

Design and implementation of a new local control system at the S-DALINAC

Vom Fachbereich Physik
der Technischen Universität Darmstadt

zur Erlangung des Grades
eines Doktors der Naturwissenschaften
(Dr. rer. nat.)

genehmigte

D i s s e r t a t i o n

angefertigt von

Dipl.-Phys. Simone Georgia Richter
aus Berlin

Darmstadt
D 17

Referent: Professor Dr. rer. nat. Dr. h.c. mult. A. Richter
Korreferent: Professor Dr.-Ing. T. Weiland

Tag der Einreichung: 6. Dezember 2000
Tag der mündlichen Prüfung: 12. Februar 2001

Zusammenfassung

Im Rahmen der vorliegenden Arbeit wurde das lokale Steuersystem des Darmstädter supraleitenden Elektronenlinearbeschleunigers S-DALINAC neu entwickelt. Das in der Aufbauphase des Beschleunigers eingeführte verteilte Kontrollsystem war in seiner lokalen Kontrolleinheit auf Basis eines 16-bit Rechners nicht mehr weiter ausbaufähig. Während des mittlerweile über zehnjährigen Betriebes ergab sich die Notwendigkeit, die Strahlführung mit neuen Komponenten zu erweitern und weitere Funktionen in der Steuerung des S-DALINAC einzuführen. Diese Verbesserungen sollen auch zur Erhöhung der Reproduzierbarkeit der Strahleinstellungen dienen.

Das neue lokale Steuersystem wurde aufbauend auf zwei 64-bit AXPvme VMEbus-Rechnern entworfen und implementiert. Die Einbindung der zur Strahlführung notwendigen Hardware erforderte eine Neuentwicklung der Schnittstellen für die im Institut gebauten Magnetnetzteile und Strahldiagnoseelemente. Das Konzept der Steuerungssoftware basiert auf dem Echtzeit-Betriebssystem VxWorks und einem, modernen Softwareentwicklungen entsprechenden, objekt-orientierten Ansatz. Erste Betriebserfahrungen mit dem neuentwickelten lokalen Kontrollsystem haben gezeigt, dass die Ziele – eine größere Reproduzierbarkeit der Strahleinstellungen und eines erweiterten Funktionsumfangs – erreicht worden sind.

Contents

1	Introduction	1
2	Remote and local control system at the S-DALINAC	3
2.1	The electron accelerator S-DALINAC and its beam transport control system	3
2.1.1	ACP	8
2.1.2	VIEW	9
2.2	Former implementation of the local control system for the S-DALINAC . .	9
2.3	Limitations of the former control system	12
3	Requirements of a modern accelerator control system	14
3.1	Soft- and hardware for local control systems	15
3.2	Modern software engineering methods	18
4	Design concepts and implementation of the S-DALINAC local control system	20
4.1	Hardware developments	20
4.2	Software design and implementation	21
4.2.1	Device groups	24
4.2.2	Task structuring	26
4.2.3	Task interactions	28
4.3	Consequences for the S-DALINAC remote control system	31
4.3.1	Grouping of magnets	31
4.3.2	Cycling of magnets	32
4.3.3	Influence onto saving and restoring of settings	36

5	Operational experience	37
5.1	Implementation and testing phase	37
5.1.1	Code implementation and testing	37
5.1.2	Commissioning of the new S-DALINAC local control system	39
5.2	Experience in daily use	40
6	Perspectives	41
A	Comparison of accelerator control systems	44
B	S-DALINAC control system	46
B.1	Hardware overview	46
B.2	Software overview	50
C	System installation and upgrade	52
C.1	AXPvme Single Board Computer	52
C.2	System upgrade	53
D	Inter task communication tools used at the S-DALINAC local control system	55
E	Q-bus-VMEbus converter	58
F	Device drivers under VxWorks	62
F.1	Driver entry points	62
F.2	VMEbus drivers under VxWorks for AXPvme	62
G	LCP implementation under VxWorks	65
G.1	Ethernet hooks	65
G.2	Protocol extensions	66
H	New ACP commands	68
	Glossary	70
	Bibliography	75

Chapter 1

Introduction

Since the 1960s the investigation of nuclear structure is one of the main research topics at the Institut für Kernphysik of the Darmstadt University of Technology. Until 1988 the normal conducting DALINAC (Darmstadt linear accelerator) [1] provided a pulsed electron beam with energies between 30–70 MeV. To access additional experimental areas, namely performing coincidence experiments, the superconducting Darmstadt electron linear accelerator S-DALINAC [2] was proposed in 1982. It provides a high quality low emittance continuous wave electron beam with energies ranging from a few MeV up to 130 MeV and currents up to 20 μA . The acceleration system uses superconducting cavities operating at a frequency of 3 GHz, whereas the beam transport system relies on normal conducting magnets delivering the electron beam to the different experimental sites located either in the accelerator hall or in the experimental hall.

While the DALINAC was operated with an analog control system, the S-DALINAC with its multitude of devices demanded computerized control systems. Two independent ones, one for the beam production and acceleration system and one for the beam transport and diagnostic system, were designed and implemented during three doctoral [3–5] and many diploma theses. Due to limited financial resources, cost-effective control systems had to be developed. This implied that most of the hardware components were built by the electronics workshop of the institute. To integrate these components, the control system software was specially designed for.

Since the design of the rf acceleration system has not been changed over the years of operation, there was no need to substitute this local control system. In contrast to that, the beam transport system was extended and altered continuously. Thus the corresponding beam transport control system needed to be modified to reflect these changes. The former system was no longer expansible because:

- With additional hardware to control, the memory space of the control systems' computer was exceeded. New devices could not be administered in the computers' device database.
- The amount of serial line interfaces was exhausted. However magnet power supplies for new beam lines had to be connected using serial line interfaces.

- Due to the increased operational experience how to set up an electron beam, the operators requested new functions like the cycling and grouping of magnets.

That was why the local control system had to be replaced in software and partly in hardware. Some hardware components, like power supplies, view screens etc., could not be replaced. Therefore control systems like EPICS, the Experimental Physics and Industrial Control System [6], or control system for the CERN or GSI accelerator sites could not be employed at the S-DALINAC since they strongly depend on the existence of standard hardware interfaces.

The structure of the former local control system influenced strongly the implementation of the remote operator interfaces. They should not be completely revised, only be extended with the new functions the new local control system offers. The work carried out during this thesis focused on the design and implementation of a new local control system software based on the concept proposed in [4] which consists of an Alpha processor based VMEbus Single Board Computer AXPvme and the real-time operating system VxWorks. Nowadays all modern accelerator control systems follow an object-oriented design approach. With this approach a well defined model of the accelerator and the functions the control system has to provide is created, which ensures easy maintainability and upgradeability during the lifespan of the machine. Therefore this thesis focused on the object-oriented modeling of the S-DALINAC control system. Modern software engineering methods have been applied to guide the encoding process and to lead to a robust and scalable system.

This thesis is divided into six chapters. The accelerator S-DALINAC is presented in the first part of Chapter 2, whereas the second part introduces the existing remote applications for the beam transport and diagnostic systems. It finishes with a discussion of the former local control system with its limitations. Chapter 3 focuses on the components of a modern accelerator control system and modern software engineering methods. The design and implementation of the new local control system is the content of Chapter 4, which also includes the description of the modifications of the remote operator interface. The commissioning of the new control system together with an overview of the first daily experiences is described in Chapter 5. The perspective (Chapter 6) for the overall S-DALINAC beam transport system closes this thesis. The first appendix (Appendix A) compares the S-DALINAC control system with other existing ones. Appendices B–H cover details of the implementation.

Chapter 2

Remote and local control system at the S-DALINAC

The local and remote control systems for the superconducting Darmstadt electron linear accelerator S-DALINAC are the outcome of a 15 year plus development. This distributed nature originates from the fact that the demands for controlling the accelerator devices are completely different from the demands for user-friendly operator interfaces. The local control system consists of very specialized applications and hardware. Thus both the control application and the supporting operating system must be tailored especially for the S-DALINAC. The remote operator applications provide tools to set up and monitor the beam. They have to reflect the structure of the accelerator devices but may otherwise depend on standard workstations. Communication between these two parts is implemented with a private protocol distributed via the laboratory's Local Area Network¹ (LAN).

For the S-DALINAC two independent local control systems are implemented: one deals with beam production and rf acceleration, the second with beam transport and diagnostics. Further description of the rf control system can be found in [5]. The operator interface for the rf control system [7] has been reworked recently. A short overview of the S-DALINAC precedes the introduction of the operator applications for beam transport and diagnostic. The presentation of the former local beam transport and diagnostic local control system leads to the discussion of its limitations. The new local control system designed during this thesis, now overcomes such limitations in hard- and software and provides new functions for the operator interfaces.

2.1 The electron accelerator S-DALINAC and its beam transport control system

The S-DALINAC is located in the basement of the Institut für Kernphysik of the Darmstadt University of Technology. The figures on the following two pages show the layout of the accelerator with all its devices to be controlled. They provide the operators with the information about the arrangement and name of the S-DALINAC devices. The names

¹For the definition of terms originating from the computer science see Glossary starting on page 70.

and positions of the devices are presented there. To get a quick overview over the distribution of the devices, each device type has its own symbol. Figure 2.1 shows all devices in the accelerator hall, the cavities of the injector and main linac are displayed too. They are just identified by a consecutive number, whereas the naming convention for the beam transport and diagnostic devices follow the definition introduced in [8]. The two letter code for the accelerator section (e.g. **I0** for the normal conducting injection) is followed by a two letter code indicating the device type (e.g. **SH/SV** for the horizontal resp. vertical steerer) and the consecutive number in the section. The view screens and Faraday cups are named after a shorter scheme, the section code is followed by a one letter code indicating the type and then the consecutive number. The beam lines in the experimental hall are shown in Fig. 2.1.

Table 2.1 lists the number and type of devices that are presently controlled by the local beam transport control system.

Tab. 2.1: Device Type, device code, and number of presently controlled devices at the S-DALINAC.

Device type	Device code	Number
Dipole magnet clusters	BM	5
Dipole magnets	BM	12
Dipole magnet trim coils	TM	31
Quadrupole magnets	QU	69
Horizontal steerers	SH	29
Vertical steerers	SV	36
Solenoid lenses	LE	3
Path-lengths	PL	2
Energy defining slit system	SL	1
View screens	R,T	60
Cameras		64
Current measurement for eight Faraday cups		1
ADC channels for set point control		64

Figure 2.1 shows the elements of the S-DALINAC control system that embodies the standard model of distributed control systems. For radiation safety reasons the computer systems controlling the devices must be outside the accelerator hall but as close as possible to the machine. The systems providing the operator interfaces are located further away. Such configuration is called client-server system. If the number of computer systems in the local controls is larger than one, the term "distributed local control system" is applied.

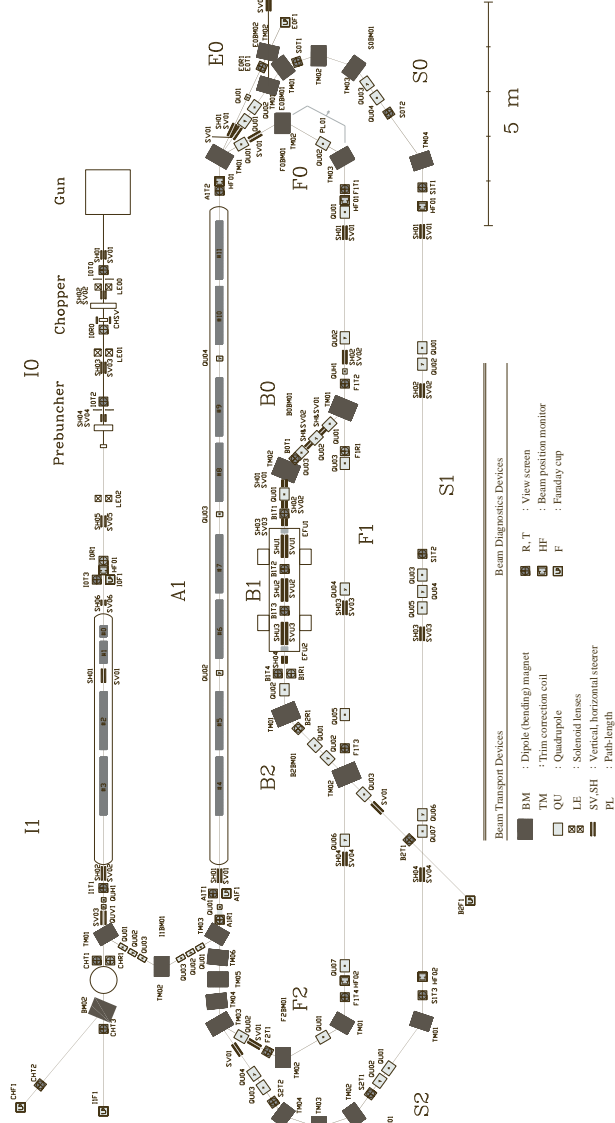


Fig. 2.1: The superconducting electron linear accelerator S-DALINAC . The electron beam is produced in the gun and the electrostatic preaccelerated beam gets its time structure (necessary for successive acceleration in the superconducting cavities at 3 GHz) in the chopper-prebuncher-section (10) at room temperature. When leaving the injector (11) the beam has an energy of up to 10 MeV and it can either be used for experiments taking place in the CH-section or bent by 180° for injection into the main linac (A1). The main linac can be used up to three times, the electron beam is recirculated twice (F0-F2, S0-S2). In the bypass (B0-B2) to the first recirculation the undulator of the Free-Electron Laser is located. The electron beam can be extricated (E0) to the experimental hall which is shown on the next page.

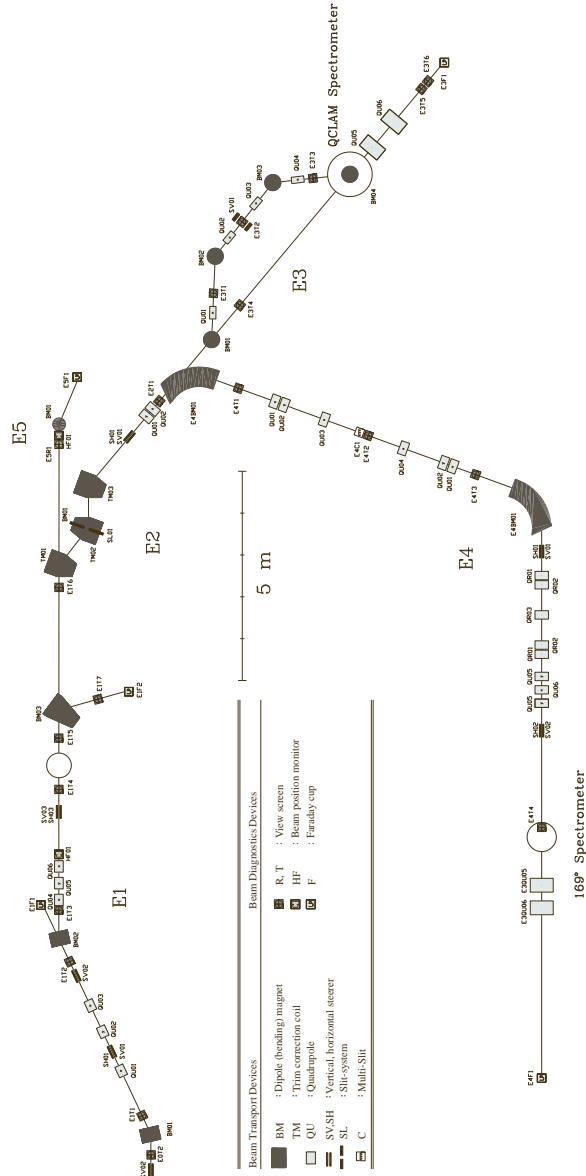


Fig. 2.2: The experimental hall of the S-DALINAC. The beam-lines transport the electron beam to the various experimental sites. High energy radiation physics experiments are performed in section E1, the magnetic spectrometers in section E3 (QCLAM spectrometer) and E4 (169° spectrometer) are instruments to investigate the nuclear structure. The energy defining 40° system is located in section E2. The polarization of the nucleus is studied in section E5.

The **O**perator **I**nterfaces, **OPIs**, are a collection of user-friendly tools used for data input or monitoring purposes executed on a workstation. In the client-server model those interfaces are called clients. The **A**dvanced **C**ontrol **P**rogram, *ACP* is responsible for the beam transport [4] and *VIEW* for beam diagnostics and control of the Faraday cups [9]. The *ACP* is a multipurpose OPI also including the function of the *VIEW* application. In normal operation only the functionality to access the beam transport devices is used. For the control of the view screens and cameras an application with a graphical user interface, like *VIEW*, is better suited. Remote access to the beam production (gun) and acceleration devices (cavities) is provided by the *HF* operator interface application [7]. The operational parameter settings of the S-DALINAC gained with the *ACP* and *HF* are stored on mass storage devices of the institute's OpenVMS-Cluster.

The servers, named **I**nput **O**utput **C**ontrollers, **IOCs**, are industrial bus-based microprocessor systems with various attached modules for digital and analog signals. The local control for gun and rf acceleration control is based on a Motorola 68020 processor VMEbus system [5]. The control interface for each cavity is provided by an analog low frequency circuit board. An overview of the IOC configuration for the former beam transport control system is given in Chap. 2.2 and a detailed explanation of the new configuration can be found in Chap. 4.1.

The connecting link between client and server is the Local Area Network, establishing transparent communication between the clients and the servers. Reflecting the special requirements of the S-DALINAC control system hardware, a private network protocol was developed and implemented by [3, 10]. The **L**inac **C**ontrol **P**rotocol, **LCP**, supports multiple receivers in the heterogeneous computer environment. The requirement for mul-

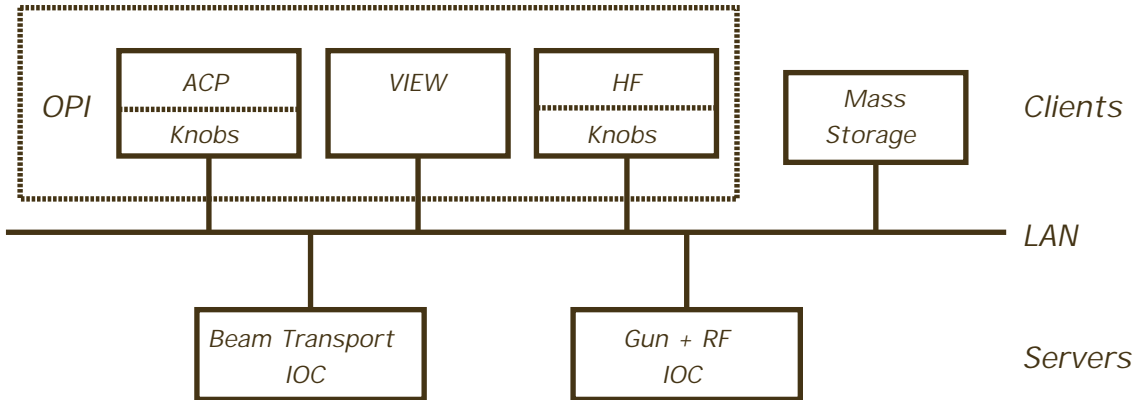


Fig. 2.1: Generalized view of the S-DALINAC control system. The upper part representing the client side shows the two operator interfaces for the beam transport (*ACP*) and diagnostics (*VIEW*) and the rf acceleration system (*HF*). The applications *ACP* and *VIEW* use the knobs as operator input devices. The lower part displays the servers which are Input/Output controllers (IOC) for the two distinct local control systems. The Local Area Network (LAN) interconnects the whole system.

multiple receivers originated from the demand to limit the number of packets that have to be sent on the LAN. Otherwise the update information of a status change in the local control system would have to be distributed to each single node involved in the remote control applications.

2.1.1 ACP

The main purpose of the *ACP* operator interface is to access the magnet power supplies and to provide mechanisms for saving and restoring accelerator settings. A command line interface is used if the operators need to directly set an absolute value of a power supply. After the electron beam is prepared the operators need to change the settings of a device in incremental steps. That type of input is done with the knobs. They provide a means for that type of operator input, where a digital value is entered through an analog device. The knobs are interfaces to the knob-server, which is a special development of the Institut für Kernphysik running on a Motorola 68020 processor. The knob-server supports up to eight so-called knob-boards each carrying four rotary knobs. Each knob-board can be assigned to an application and then each knob to a control unit. The knob-server can be accessed by the control system through an Ethernet interface but can only operate in conjunction with *ACP* or *HF*, see Fig. 2.1, because all data in the knob-server must be loaded through them.

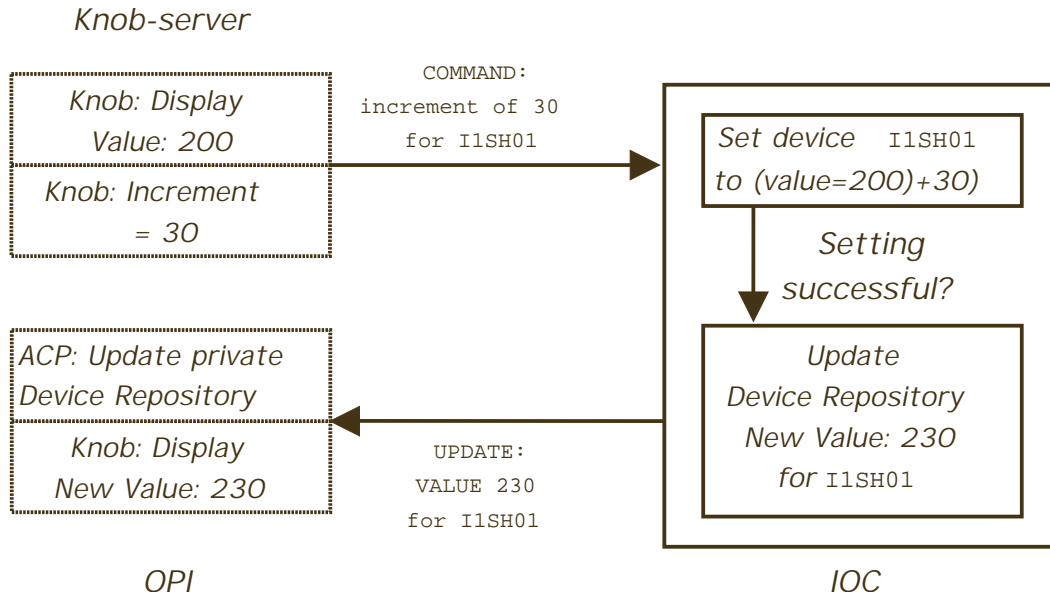


Fig. 2.2: The data flow exchanged between the *ACP* and the local control system is illustrated for the example of setting a magnet to a new value. The dotted line boxes represent actions on the client side, the others actions in the server.

The data flow between the operator interface application *ACP* and the local control system is exemplary illustrated in Fig. 2.2; it is similar for the rf control system. Based on the assumption that the operator has used the knobs to select the magnet (e.g. the steerer I1SH01) he wants to change the value for, the increments of the knob turns are evaluated by the knob-server and it sends the value of the increment together with a device identification to the IOC where the command is evaluated and carried out. After a successful setting of the new value, the parameters of the device are updated in the device repository in the IOC. Then the operator interface applications and knob-server are informed simultaneously of the change of the device setting. The display of the device value is not changed in the knob's status line until the receipt of this update.

To take some of the strain from the local control system, the device database maintained by the local control system is mirrored on the workstation where the *ACP* application is running. The memory space for the database in the workstation is a special type, where many applications can read information from. Therefore the local control application does not have to send updates to multiple remote applications but just to the *ACP*.

2.1.2 VIEW

The application *VIEW* is an OPI with a graphical user interface in which the present status of the view screen targets and the movable Faraday cups are displayed. The operation of the targets and movable cups can be initiated, the current measurement in any of the Faraday cups can be selected, and a scale for the current measurement can be chosen. Additionally, *VIEW* controls the connection of cameras (associated with the targets, the helium liquefier or experimental set-ups) with four screens in the S-DALINAC control room. Presently the *VIEW* undergoes a reimplementation [11].

Currently the *VIEW* operator interface reads its unit information from a text file during application startup. There is no direct link between the *VIEW*'s unit status and the actual status of this device. The new design approach will mirror the device status stored in the IOC by using the same mechanisms as the *ACP* application. Therefore reusing the definition macro [12] is possible and in case of both OPIs for beam transport and diagnostics (*ACP* and *VIEW*) executing on the same workstation, they can even share the same memory location for this database.

2.2 Former implementation of the local control system for the S-DALINAC

A short overview of the former local control hard- and software is presented here. To make further use of the existing operator interfaces, it is very important that the new system provides at least the same functions and communication protocol (LCP) as the

old application. It leads to the discussion of the limitations of the system as it was present at the beginning of this thesis. Very detailed description of the hard- and software can be found in the doctoral theses [3, 4] and the references therein. The former system was based on a PDP11 computer system operating under RSX-11M-Plus. It was nearly impossible to employ an existing accelerator control system of another accelerator site because those systems rely on standard hardware. Since many components of the S-DALINAC were build by the electronics workshop of the institute the amount of work to adapt another system would have been of the same volume as programming an individual S-DALINAC control system. The operating system RSX was selected because an expertise and experience was already gained while building up a data acquisition system [13].

Local control software. The former control system’s architecture follows rudimentarily the layout of an operating system. All components are implemented on a single PDP11.

Figure 2.3 illustrates the abstract view of the former local control system software. Core of the application was the central **STEUER** task and the common block **STCOM**. Later improvements and enhancements were provided by the Ethernet communication providing the **ETHSTE** task and various user programs. The **STCOM** stored all information about the device controllers (DCBs) and attached units (UCBs) in the global memory providing a device database. The content and size of this database had to be defined at compile-time and was fixed throughout program execution. There was no protection

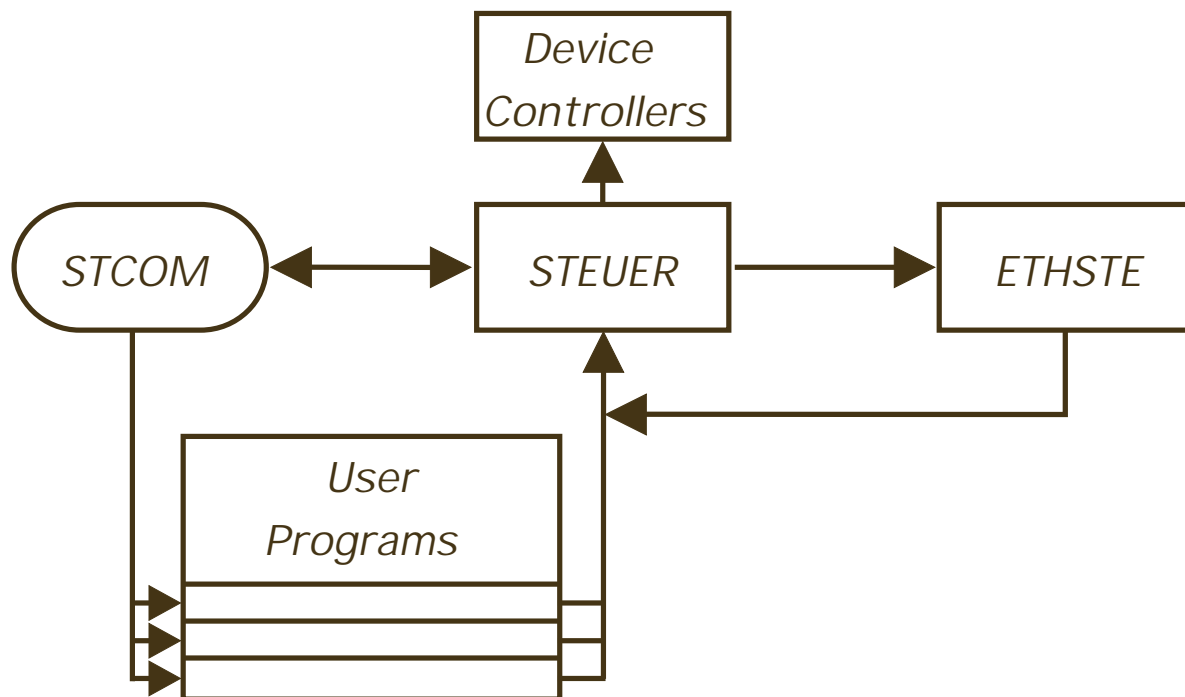


Fig. 2.3: Architecture and data flow of the former local control software (see text).

against multiple writers but the system was designed that only **STEUER** did the actual update of the common block (illustrated by the double sided arrows). The **User Programs** had implemented read only access to the common block (indicated by the single sided arrows from the **STCOM** to the **User Programs**) to guarantee data integrity. The Ethernet communication was handled in an isolated task using the same message queues to communicate with **STEUER** as the **User Programs** that dispatched the commands to the device controllers. The parts of code dealing with the beam transport devices communication did not use the RSX I/O system, but addressed directly the device controllers on the Q-bus.

Local control hardware. The central component shown in Fig. 2.4 is the CPU (type PDP11/73) forming the computer with the memory, harddisk, Ethernet interface and serial lines. Beam transport and diagnostics elements are connected to the computer via a special Q-bus extension, developed especially for the S-DALINAC. Except for some of the magnet power supplies which have a serial line interface, all devices listed in Tab. 2.1 are addressed via the Q-bus extension. The Q-bus has a multiplexed 22 bit wide address bus and a 16 bit data bus and it allows a maximum transfer rate of 3 Mbit/s.

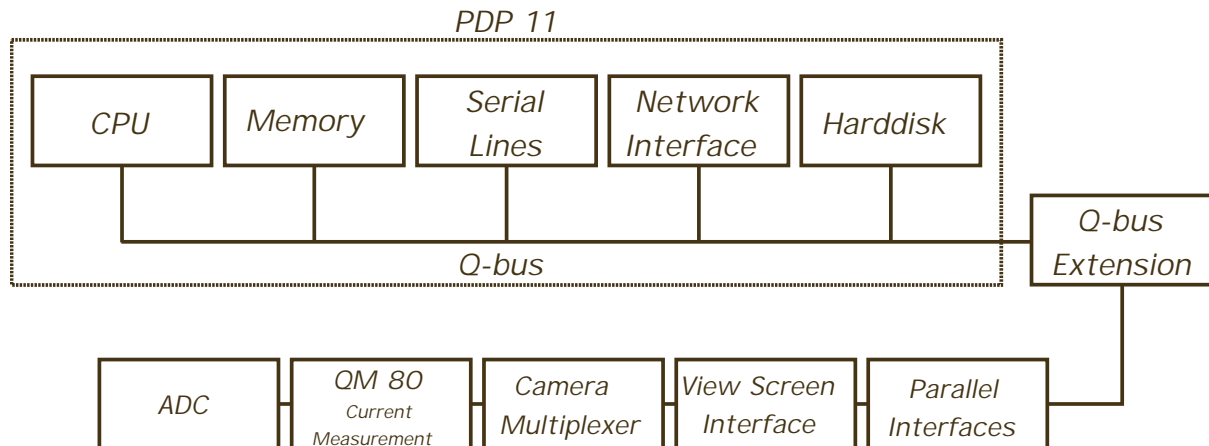


Fig. 2.4: Schematic overview of the former S-DALINAC local control hardware. Shown is the central processor (PDP11) with its internal components – indicated by the dotted line box – and its I/O modules that provide the interfaces to the beam transport and diagnostic devices on the extended Q-bus.

Figure 2.5 shows the hardware components attached to the Q-bus extension. All device controllers are directly attached daisy-chained to the Q-bus. Succeeding to the controllers their specific devices are also on a parallel device bus. The parallel interfaces are the ports for those power supplies that do not have a serial line interface. Each of those interface cards carries two channels. The crate hosting the parallel interface cards is capable to be filled up with 32 cards. Three of those crates are used. The current measurement device "QM80" operates as a multiplexer and is thus able to measure the current deposited in one of eight different Faraday cups and each in nine different ranges. To support the operators

with information about the beam quality, view screens and cameras to catch the pictures are available throughout the accelerator beamline. Three camera multiplexer and four view screen modules provide control over up to 64 cameras that can be assigned to four monitors. The ADCs provide a set-point control for the power supplies controlled by the parallel interface cards.

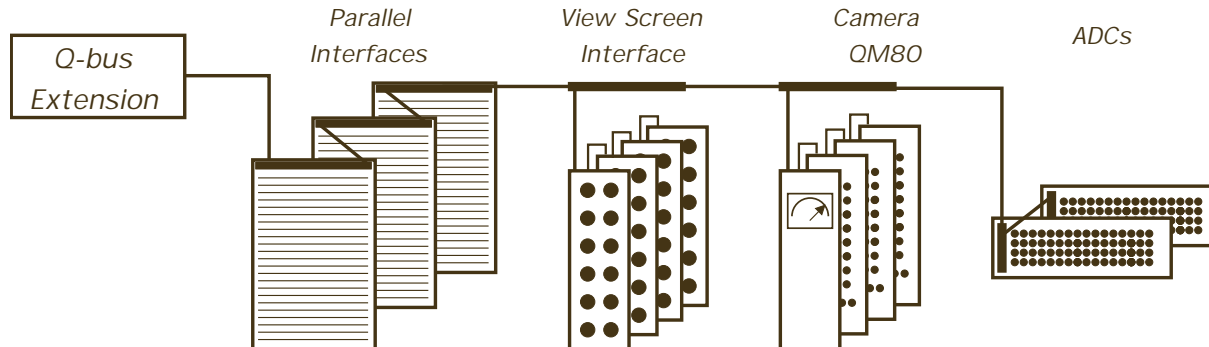


Fig. 2.5: Detailed view of the daisy-chained device topology connected to the Q-bus extension of the former local control system processor. The thick solid lines represent the controller cards for the devices.

2.3 Limitations of the former control system

During the successful operation of the S-DALINAC the knowledge base and operational experience grew tremendously. Simultaneously the number of diagnostic and set-point control elements increased. New beam lines for additional experiments, like the one for the FEL [14], and the need to integrate the beam line leading to the 169° spectrometer [15] enforced a new hardware concept for the local control system. The beam line to the 169° spectrometer already existed but is now controlled via a new interface type [16]. There are two main reasons requiring a new system. First, there was a limitation in the size (eight kbyte) of the common block **STCOM** containing the device database, which is a special feature of the operating system RSX. Second, the number of serial line interfaces could neither be increased. Without the additional interfaces the control over completely new beam lines (e.g. the beam line to the experimental site for the investigation of the polarizability of the nucleon [17]) would not have been possible to implement. Additionally the local hard disk attached to the computer system in the klystron room often caused system failures, by being destroyed by the vibrations resulting from the nearby compressor of the helium liquefier.

During the last years the operators requested several new functions to further improve the operation of the S-DALINAC. The major objective layed on reaching a higher reproducibility for the beam transport device settings but further improvements were also requested. These were:

1. **Reproducible magnet settings.** Due to the fact that some magnets have an iron core, hysteresis effects arise. The operators only have access to the set value of the current that flows through the coils but not to the exact value of the magnetic field. Affected by this hysteresis effect are the Danfysik quadrupoles and the bending dipole magnets.

Depending on the former settings of the magnet a restored setting does not lead to the same beam transport properties as they were at the time when the setting was saved. A procedure had to be defined to ensure the reproducibility of the magnetic field. The former local control system software could not provide the needed complexity level to carry out these procedures.

2. **Device groups.** After implementing a FODO-lattice structure in the straight part of the first recirculation [18] of the S-DALINAC the operators formulated the demand for a grouped device. During normal operation the six quadrupoles forming the FODO-lattice have to behave as a single device. During beam set-up the operators need individual access to each quadrupole. A dynamic grouping and degrouping of beam transport devices could not be implemented.
3. **Support of multiple Input/Output Controllers.** To reduce the amount of cabling needed to control the devices in the experimental hall, it would have been an option to install a second IOC in this hall. The former control system system was not designed to support more than one IOC. Being already at its limits the control system was no longer expandable and scalable.

Chapter 3

Requirements of a modern accelerator control system

Modern accelerator control systems must fulfil two main requirements:

- The operators have to be supported by a user-friendly interface to efficiently set up the beam and deliver it to the experimental sites.
- The application developer has to rely on an open, well structured system. The system design should incorporate modern software engineering techniques. These techniques force building a more reliable and well maintainable control system. This is especially helpful when the supporting staff is changing during the life span of the accelerator.

The cooperation of the remote control system, a collection of operator interfaces, and the local control system, consisting of Input/Output controllers, forms the total control system. This concept of a distributed system with different responsibilities among the components allows flexibility to a high degree. The local control system has the duties to steer, monitor, and administer the accelerator devices. It must be capable to run independently from the remote control system ensuring access to the accelerator even in the case of a failure of the remote applications. The whole system should be recoverable with defined parameter settings. After system failure that led to a loss of the local control, the control system must provide an opportunity to restore the original state. In contrast to the local control system that must have full knowledge about all devices and their parameters, the remote control system works with a subset of the device database. Only the device information relevant for the specific application must be mirrored in the local control system. Further data processing, that does not directly influence the beam control, can be executed on the remote computer systems. Thus the local systems are freed from time consuming operations, ensuring faster response times.

The main characteristics of early control systems, originating from the 1980s, is their tight interconnection between the remote and local control system. Modern system design uncouples both systems leading to more independent systems. This approach enables the easy adaptation to increasing demands either in hardware (e.g. more devices) or in software (e.g. new functions).

Since the purpose of this thesis was the implementation of a new local control system the following subchapter presents modern hard- and software components especially for Input/Output controllers. The discussion of modern software engineering methods in Chap. 3.2 is applicable to both the remote and local control system.

3.1 Soft- and hardware for local control systems

To provide a continuous service of the beam transport the supporting operating system software of the local control system [19] has to fulfil tight timing requirements. The operating system must support mechanisms for parallel handling of many events. Thus it must provide at least the following two features as there are:

- concurrency
- real-time behavior

Concurrent systems. While operating an accelerator many different duties may be performed simultaneously. Several periodic duties, like set-point control, saving settings or communication between the IOCs and OPIs, are performed parallel to the processing of operator input. Structuring the application in concurrent tasks results in a separation of concerns of *what* each task does, from *when* it does it. This usually makes the system easier to understand, to construct and to manage.

As these tasks cannot be treated completely isolated since many of them depend on actions or events generated in others, several tools of inter task communication [20] are provided by modern operating systems:

- semaphores
- signals
- pipes
- message queues
- shared memory

Appendix D describes in detail these inter task communication tools. A comprehensive understanding of them is necessary to implement or modify the accelerator control system application. These tools work on the lowest layer of the operating system, so that misuse may easily lead to a failure of the system.

Real-time systems. Operating systems that are compliant to the POSIX P1003.13, *Draft 5 Minimum Real-time System Profile* [21] are called real-time systems. They must react within a deterministic time span. From a practical point of view, a real-time system could be seen as a concurrent system with tight timing constraints, having to process different types of input and to produce multiple outputs. Inherently control purposes must rely on the compliance of reaction times and actions to all kinds of events. The minimal time span that is inherent to the system reacting to an event is called latency. These latency times are depending on the underlying hardware and the operating system. For a real-time system further delay in event processing (time spans greater the latency) may result in a fatal failure. When the operators change values of magnet power supplies they should have the impression that the command is carried out instantaneously. This is given when the latency time is less than 20 ms. Actually response times are much faster (in the micro second range) but at the S-DALINAC they are artificially delayed (15 ms) since the knob-server hardware cannot deal with high rate (μ s) updates. A hardware damage would be the result of high rates.

Internally the local control system depends on short latency times so that the features of priority preemptive scheduling and interrupt handling is mandatory. In a priority based multi-tasking system each process is assigned a priority and the process ready to run with the highest one is assigned the CPU. If, as shown in Fig. 3.1, a higher priority task T3 changes its state from **pending** to **ready**, the kernel will immediately switch to the context of the higher priority task, after having saved the context of task T2. This procedure is called **preemption**. The execution of task T1 with the lowest priority will be delayed until the other two tasks T2 and T3 do not need the CPU any longer.

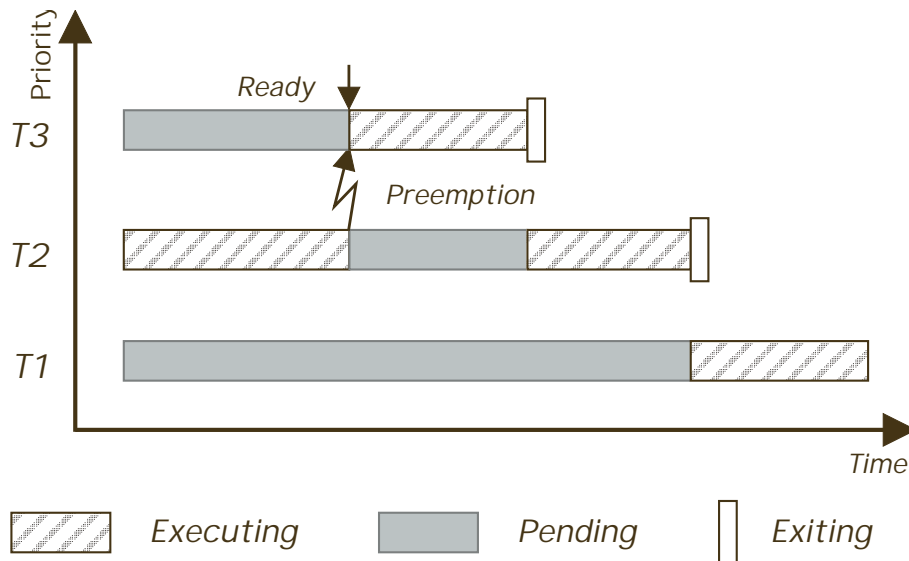


Fig. 3.1: Preemptive-priority scheduling.

The components of an accelerator control system fulfilling these requirements are presented in the following. An operating system must be compliant to the above specifications, one of the commercially available systems is VxWorks. To build up an accelerator control system that is flexible enough to incorporate many different devices should be compliant to an industrial standard bus, like the VMEbus. It interconnects the computer with the device controllers. An essential part of the control system is the LAN that links the remote to the local control system.

VxWorks. The special real-time operating system VxWorks [22] is employed in many accelerator control systems. For example the multipurpose Experimental Physics and Industrial Control System [6] relies on VxWorks. It fulfils the above specifications for a concurrent real-time system and is capable to operate without a hard disk. It has UNIX-compatible networking capabilities and forms together with UNIX or Windows NT a hybrid development platform fully relying on a host-target development environment, illustrated in Fig. 3.2. The host provides development support through standard software such as editors, cross-compiler, linker or non-real-time debugging facilities, while the VxWorks target just executes the real-time application standalone. After compilation and linking of the application on the host system, it is downloaded via the network link into the target. During the operation of the local control system the network link between host and target is not required.

The operating system can be tailored to exactly meet the specifications of the application. Various modules can be added by compiling them into the system image.

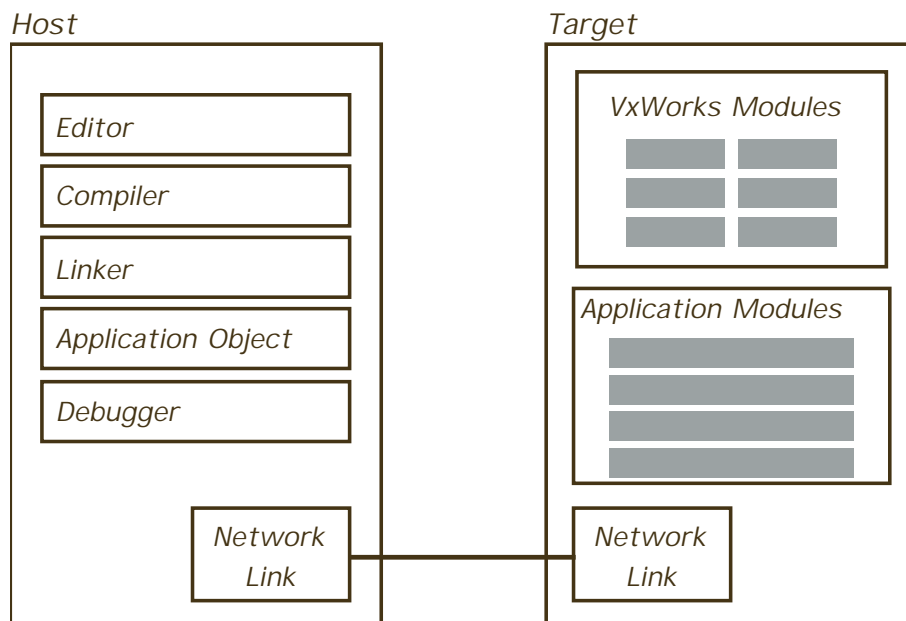


Fig. 3.2: Host-Target environment for VxWorks. The network link is provided by the laboratory LAN.

VMEbus systems. The interaction with the accelerator devices is the main objective of a local control system. Bus systems provide this interconnection between the Input/Output controllers and the attached peripheral devices. A very popular bus is the VMEbus. Because of its open standard [23] and mechanical robustness it is widely spread, and one can find many manufacturers offering VMEbus components. This allows to build the hardware system to exactly meet the demands of the accelerator. The basic specifications [24] are:

- Addressing capability up to 4 Gbyte
- flexible transfer modes (programmed I/O or faster DMA)
- coordinating the bus arbitration through the system controller
- providing 7 interrupt levels to initiate a priority-based fast data exchange

Ethernet Networks. A mandatory component of a distributed control system like an accelerator control system is a reliable network link connecting the remote operator interfaces with the Input/Output controllers. The most popular network protocol in LANs is Ethernet. The latest specifications of the IEEE [25] allow transfer rates up to 1 Gb/s, the network diameter is depending on the transport media and ranges from 100 m up to 5 km when using mono-mode fiber optic cables.

There are three different possibilities to address nodes on the network:

- **Unicast** – a single node is addressed
- **Multicast** – a group of nodes is addressed
- **Broadcast** – all accessible nodes on the same domain are addressed

Especially the second addressing mode is used in accelerator control systems for the communication from the Input/Output controller to the operator interfaces. It allows fast and simultaneous updates of accelerator status changes that have to be propagated. A unicast addressing is used when the operator interface sends out a command to an Input/Output controller.

3.2 Modern software engineering methods

An extensive requirements analysis precedes the modeling process of the accelerator control system applications. The model brings together the hardware specifications and the requirements formulated by the operators and accelerator physicists. To transform this model into code, object-oriented methods have been used since the early 1990s. Especially

control systems for large accelerator installations, like those at CEBAF [26] or DESY [27], rely on those modeling techniques, since otherwise the complexity of the different control activities is no longer manageable.

The object-oriented approach leads to a more abstract model of the control system application. This allows to use common problem solution strategies, called design patterns [28], which are structural independent from the underlying specific problem. Since an accelerator undergoes constant improvements its control system has to be adapted to those changes as well. The definition of independent patterns and encapsulation of data in well defined modules, called information hiding [29], increases the flexibility and extension of the control system. Since different modules use specific interfaces to other modules only they do not need to be changed if one component is replaced. This advantage allows the developer to respond rapidly to demands caused by accelerator changes or new operator requirements while still maintaining the overall stability of the control system and quality of the application code.

It is not necessary to implement the control system application in an object-oriented fourth generation language like C++ [30] or Java [31]. The above methods are independent from any programming language. The third generation language C is suitable to develop object-oriented local control systems with the interesting aspect that it is widely known among physicists.

Chapter 4

Design concepts and implementation of the S-DALINAC local control system

This chapter is organized in three subchapters. The first section deals with the development of the new hardware components. This development was necessary to incorporate the existing beam transport devices and diagnostics elements into the new local control system. The presentation of the fundamental software design decisions and how they were implemented is given in Chap. 4.2. The new functions that are now provided by the local control system must now be reflected in new functions in the operator interfaces. Those improvements of the remote control system are described in the last subchapter.

4.1 Hardware developments

The new hardware concept for the S-DALINAC local control system was proposed in the thesis of [4]. The concept also included the choice of the supporting operating system and the computer type. Thus the full IOC configuration was specified:

- the industrial bus system VMEbus as successor to the Q-bus
- the real-time operating system VxWorks as successor to RSX
- the Alpha processor based Single Board Computer, AXPvme SBC, as successor to the PDP11

In spite of the transition from the Q-bus to the VMEbus the existing power supply interfaces for the beamline magnets as well as the view screen controller, camera multiplexer and the QM80 controller had to be incorporated into the new hardware. Their interfaces were adapted to the Q-bus extension of the RSX system. Therefore cost-effectiveness required not to replace approximately 190 magnet power supplies, but to implement a Q-bus-VMEbus converter for the new system. The Q-bus system implemented a daisy-chained bus topology. The Q-bus signals were passed through the attached controller cards. A break in this chain led to failure of the whole computer system. The new system

should not show this behavior. A failure in one of the controllers should only affect the devices controlled there and not the whole system. Therefore a distributed star topology was designed for the new hardware system. The VMEbus distributes the signals in parallel to all cards in the VMEbus crate. In contrast to the former system where only one controller card was needed for a set of devices, see Fig. 2.5, now a pair of cards, see Fig. 4.1, is required:

- Q-bus-VMEbus converter card
- controller card

Preliminary work on the pair of controller cards were already begun before the start of this thesis [32]. The final formulation of the logic equations for signal generation and layout of the printed circuit board was defined and build during this thesis, see Appendix E.

Each parallel interface crate gets its own Q-bus-VMEbus converter associated with a crate controller. The type of signals needed to control the view screens, cameras and QM80 are the same and each of those controller cards is able to control up to eight modules. Thus the QM80 and the camera switches are combined to be controlled by one single controller. The decision was made not to keep the former ADC modules but to rely on commercially available hardware. At the moment four 12-bit ADCs each with 16 channels are incorporated in the system, but the number will be extended to provide a set-point control for all of the power supplies. The AXPvme SBC is only equipped with one serial line serving as console line. Magnet power supplies with serial line interfaces are connected to the VMEbus using a commercially available carrier board that can be configured to provide up to eight RS-485 serial line interfaces.

During the testing phase of the overall system an incompatibility between the commercially manufactured modules and the converters developed on site was detected. Therefore the modules had to be inserted in separate VMEbus crates.

The modular software concept, which is presented in the following section, already enabled the use of more than one Input/Output controller. Thus the set up of a second Input/Output controller, called slave IOC, was carried out straightforward and finished within a week. Figure 4.1 shows only the configuration for the master IOC, the slave configuration follows the conventional design of an VMEbus system. Table 4.1 shows the distribution of the different plug-in modules over the two IOCs.

4.2 Software design and implementation

During the design of the S-DALINAC local control system the object-oriented methods and guidelines were consequently employed [33]. The outcome of the requirements analysis phase were the following items:

Tab. 4.1: Distribution of the VMEbus plug-in modules in the IOCs.

Master IOC	3 Parallel channel interface crate controllers 1 Camera/QM80 crate controller 1 View screen crate controller
Slave IOC	1 I/O module carrier board equipped with serial I/O modules 1 14-bit ADC for beam diagnostics 4 12-bit ADCs for set point control

- Providing a central repository for all device information
- Communication with the remote control system application via LCP
- Supporting composite devices
- Providing an interface for automated actions
- Providing the extension for multiple IOCs

Figure 4.2 sketches the structure of the layout of the new S-DALINAC local control system. The main components are the dispatch task and the device database. The symbol for the dispatch task encloses that of the database to emphasize their tight binding. Two communication interfaces are provided, one for the Ethernet communication via LCP and the second for local command input. The command evaluating dispatch task distributes the commands among the various device handling tasks or to the special purpose tasks (e.g. the task responsible for the cycling of magnets). The devices are addressed by the

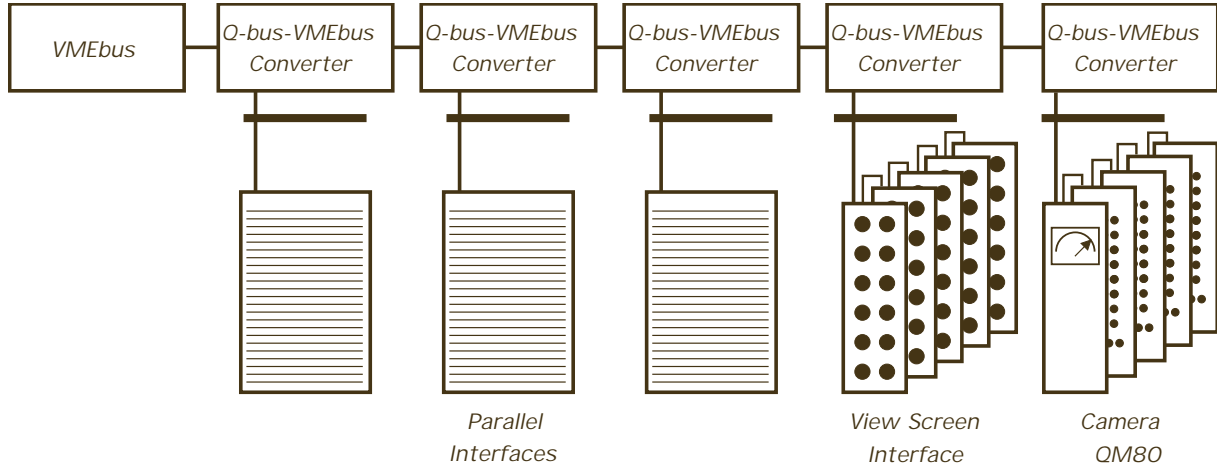


Fig. 4.1: Detailed view of the implementation of the distributed star topology of the VMEbus based system. The thick solid lines are the controller cards for the devices; each of them needs a converter card in the VMEbus crate.

device controllers, which are added to the I/O system by device tasks.

Attention must be turned to the direction of interaction represented by the arrows. The dispatch task can only read from the device database, whereas the device handling tasks are allowed to manipulate it. Since the device database will not be implemented in a task, but in global memory, access to it implies direct reading resp. writing of physical memory locations (indicated by dotted line arrows). In contrast to that, task communication (indicated by solid line arrows) is provided through message queues. These message queues resemble the interfaces between the different objects, following the object-oriented design approach. The concept of data encapsulation is implemented here by defining individual tasks for a distinctive group of operations.

The actual implementation contains two Input/Output controllers in a master-slave configuration. The operator interfaces on the remote side do not need any information about the distribution of the devices since they communicate only with the master IOC. This approach was mandatory to be able to implement multiple IOCs without modifications of the remote control applications. Both IOC systems, build up with nearly identical code, are distinguishable through the device drivers for the attached devices and the use of the device database. This database is the core of the local control system, reflecting the current status of the S-DALINAC. It stores, next to the static device attributes, the dynamically alterable attributes of the beam transport and some of the beam diagnostics units. Only the master IOC has access to the complete database, the slave IOC only knows about its physically attached devices. The below figure does not illustrate this fact, since this is not of importance until the actual implementation is carried out.

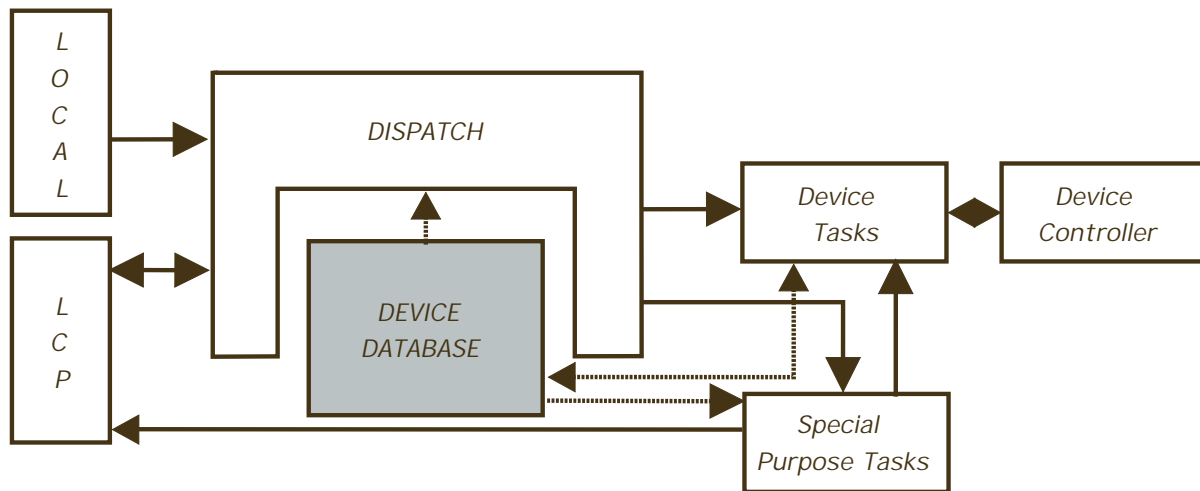


Fig. 4.2: Shown are the generalized objects forming the control system application. The database (in gray) is the only object that is not implemented as a task. The arrows represent the direction of the flow of information. Solid line arrows illustrate message queues, dotted line arrows direct memory access.

Following the concept of data encapsulation, where each functionality should be reflected in a separate task. However the handling of the accelerator device database deviates from that concept. The database can neither be associated with a single module nor can it make up a module itself as no direct functions are apparently connected to it. Different functions out of some of the modules can read from the database and are allowed to update it. To accomplish the flexibility of the code the database is created as a double linked list, which can be accessed by any task in the system. Active semaphore, see App. D protection is implemented to ensure data integrity and protection against multiple writers. If two tasks would alter the database simultaneously the worst case could be a wrong setting of a device possibly leading to damage of the S-DALINAC.

4.2.1 Device groups

Closely related to the description of the device database is the issue of supporting composite devices. During the requirements analysis the specification for grouped devices was formulated. Active grouping and degrouping of devices should be initiated from the remote control. This is very helpful for the operators e.g. in the case of the FODO-lattice [34] in the first recirculation of the S-DALINAC. The master IOC has to react to the requests by dynamically building up a composite device structure. During the testing and commissioning phase of this local control system an additional form of composite device structure proofed mandatory. Bending magnets and their correction trim coils have to be a static composite device when cycling is an issue, Chap. 4.3.2 will explain in detail the reasons for this requirement. The implementation of these two different subjects (dynamic and static device groups) must be handled transparently concerning the relay of information to the device database and also the relay to the OPIs.

The implementation of support for such composite devices demanded an upgrade of the database layout. A common coding pattern guided the programming of these two types of additional device structures. The **composite pattern** [28] composes the devices into a tree structure. It represents the hierarchies of the devices. For the remote control individual single devices and composites are treated uniformly. This allows to keep the *ACP* application almost unchanged.

Two extensions to the structure definition of the device database, reflecting the diverging needs for the static and dynamic device groups, were designed. The static composite devices are only needed when a cycling command for one of the dipole cluster bending magnets is issued. Then the root device must always be the bending magnet power supply. The power supply of a correction trim coil can be cycled independently from the power supply of the bending magnet.

Ideally the cycling of this type of static composite devices should be handled the same way as the cycling of dynamic ones. Dynamic composite devices are constructed on demand. Due to the naming conventions for the S-DALINAC beam-transport devices, the togetherness of the three 180° injector bow bending magnets and the last three out

of the chicane cannot to be derived during runtime. This information must be explicitly put into the system by the programmer at compile-time. Figure 4.3 illustrates this fact in displaying a fraction of the layout of the S-DALINAC. The magnets identified by the names of their correction trim coils I1TM01 – I1TM03 and F2TM04 – F2TM06 belong to the same power supply I1BM01.

Figure 4.3 lists the composite device of I1BM01 in the right part. For further explanation purposes the FOBM01 is included with its correction trim coils in the figure. For a dynamical building of the composite devices, it would be sufficient just to scan the database for TM unit names beginning with F. This is impossible because of the above reasons. Therefore in the device database each bending magnet has a pointer to the list of its correction trim coils, but the trims themselves rest as separate units in the database without a link of their corresponding bending magnet.

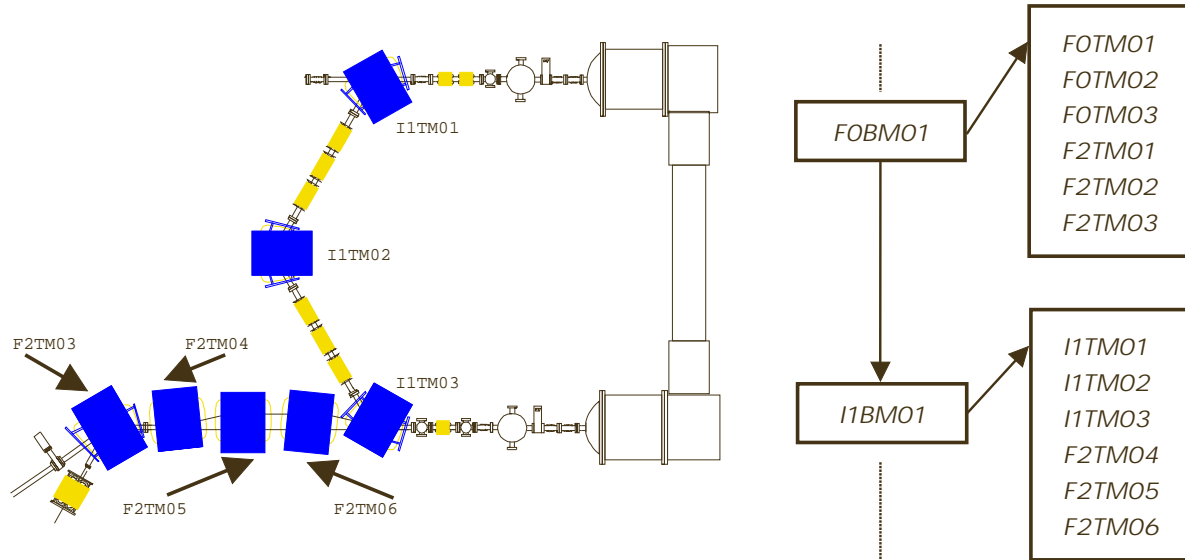


Fig. 4.3: Shown is the S-DALINAC injector bow and the chicane magnets with their names as they are addressed in the control system. Both series of bending magnets with their corresponding correcting trim coils are examples for static composite devices. The F0 bending magnet and the trims are not shown since they reside too far away from the bow.

Dynamically constructed composite devices are built up at run-time. For example of the FODO-lattice the construction of such a grouped device is illustrated. An excerpt of the actual double linked list building the device database is shown in Fig. 4.4. The root device F1QU02 has a link to the next FODO quadrupole F1QU03, which has itself a link to F1QU04 and so on. The links making up the device database (shown in the lower part) still remain, so the root device has also a link to F1QU01.

Since with the dynamic device groups the operator only sets the value of the root device,

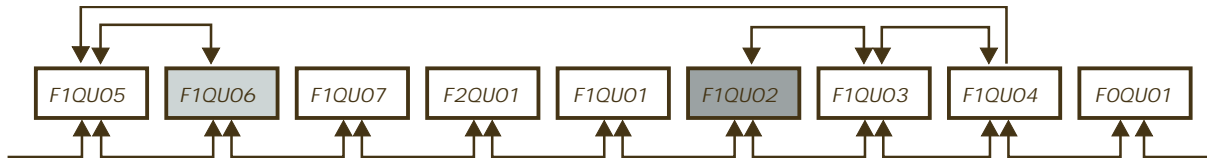


Fig. 4.4: The FODO-lattice quadrupole group, representing a dynamically created composite device is displayed. The dark grayed device F1QU02 is the root of a group formed by the F1QUxx that ends with the light grayed F1QU06. The arrows in the upper part represent the links between the grouped devices, whereas the arrows in the lower part represent the device links in the database.

the support of that feature is not only concatenating the devices to a list. Additionally the local control system must provide a database element that contains the set value ratio of the concatenated device and the roots' device set value. If a command is addressed to a dynamically grouped device the dispatch task of the local control system must duplicate the command for each group member. The set value is calculated for each single device in the group from the stored ratio and the requested value.

4.2.2 Task structuring

According to the modular design approach the S-DALINAC local control system consists of various tasks. To give an overview of these tasks, a short description of each of them follows:

Dispatch. Being the receiving task for all input (local or over Ethernet) it dispatches the commands via message queues to the appropriate device handling tasks or initiates a cycling process.

Device handling tasks. The purpose of these tasks is to install the device drivers and add the devices which they administer into the VxWorks I/O system, see also Appendix F. As the S-DALINAC control system must rely on the integrity and presence of the devices, the execution of the whole application will be stopped if these tasks cannot successfully add all devices to the I/O system.

The knowledge how to address the devices is hidden in the device drivers. Concerning the flow of command all device types are treated equally. Only the task responsible for the serial devices has implemented an additional feature. Serial communication is bidirectional and asynchronous. Therefore serial line receiving tasks are installed to avoid the blocking of the sending task when waiting for an answer from the device.

Due to the open and object-oriented design it was possible to reuse the code written for the master IOC when implementing the slave; of course the drivers had to be developed separately and the dispatch task had to be tailored for each Input/Output controller

configuration. Table 4.2 lists the different device handling tasks in the master and slave IOC.

Tab. 4.2: Device handling tasks in master and slave IOC.

Master IOC	NHDev	Management of the beam transport magnets and correction trim coils with Danfysik power supplies or power supplies developed on site.
	TGDev	Management of the fluorescent- and OTR-view screens.
	CameraDev	Management of the camera multiplexer and the QM80.
Slave IOC	SerialDev	Management of the power supplies controlled over serial lines.
	RXxx	Task concerned with receiving input from the serial lines.
	ADCDev	Management of the Analog-to-Digital-Converter Boards.

CYCLE. As already mentioned the cycling task has to initiate the correct ramping of the dipoles and quadrupoles. During this time the execution of commands concerning the setting of the involved power supplies must be suppressed. One option would have been to prevent any command execution, but this is by no means compliant to the behavior of a real-time system. Instead a semaphore inhibits all commands related to the involved power supplies.

SAVE. The save task saves the the dynamic changing portion of the device database to a file on a regular basis (every 30 s). This task prevents a complete data loss in case of a system failure because it is not possible to read the values stored in the parallel interface cards.

LOCAL. Covering the local input possibilities, this task is one of the two interfaces for the DISPATCH task. When this task is executed the console of the AXPvme SBC is used to display a list with all devices, their power status and their value, where either by cursor commands or via a command line interface device settings can be initiated.

TRANS_IN. During standard operation TRANS_IN is the main input source for the DISPATCH task. Getting the LCP packets from the Ethernet, it reacts compliant to the LCP definitions and covers – according to the OSI 7-Layer model [35] – the transport and session layer. The TRANS_IN task knows about the link status and assures the accurate transmission of Ethernet packets. Commands belonging to the application layers are passed through the message queue to the DISPATCH task.

ETHER_OUT. As the partner of TRANS_IN, this task is concerned with the outgoing communication to the network partners via the Linac Control Protocol. Concerning the LCP implementation for the TRANS_IN and ETHER_OUT radical modifications to the systems firm– and software had to be carried out. The operating system VxWorks only has native support for the Internet Protocol (IP) stack that cannot be used for LCP communication purposes.

Not yet addressed in the above discussion is the issue of task priorities. As discussed in Chap. 3.1 the assignment of different priorities to the tasks is a prerequisite to ensure that the application can meet the tight timing requirements. Table 4.3 lists the different priorities for the tasks. The assignment is the result of functional tests to deliver the best system performance. It should be noted that in VxWorks priority 0 is the highest one, and 255 the lowest. The levels are listed as they are specified in the task spawning function call. A short description why this priority was assigned is also provided.

Tab. 4.3: Task priorities in slave and master IOC. The lower the priority level the higher the actual priority.

Task Name	Priority level	Description of priority level
TRANS_IN	10	To ensure no LCP packet loss, this task is assigned the highest priority.
ETHERNET_OUT	11	Link partners rely on fast updates and responses.
TGDev	30	Controls the movable Faraday cups among the targets and should react as fast as possible to ensure machine security.
DISPATCH	40	Triggers the device handling tasks.
Device handling tasks	50	All device handling tasks can run on the same level, as no device is preferred.
RXxx	60	Serial receiving is mostly done in interrupt mode, controlling actions can be delayed.
CYCLE	80	Not time critical, can run in the background.
SAVE	100	Not time critical, can run in the background.

4.2.3 Task interactions

The process of task structuring focused on the distribution of functions to tasks. The communication and interaction between these tasks were only rudimentarily specified. The complex of task interaction in a multi-tasking environment is crucial and has to be discussed more detailed as it is mandatory to efficiently control the accelerator. The

overall S-DALINAC control system (remote and local) is based on passing information and commands from one application to another. The local control system makes extensive use of this method. This behavior is reflected in the **chain of responsibility** pattern [28]. The S-DALINAC beam transport control system does implement various chains in different lengths that avoid a direct coupling of sender and receiver. The remote applications do not need the information if the beam transport device is administered by the master or slave IOC. On the receiver side more than one task is given the chance to handle the command by passing the request along the chain until a responsible task executes it. Receiving tasks share a common interface (message queues) for handling the requests as well as passing them to their successor. This behavioral pattern characterizes the complex flow of control and allows a great flexibility in distributing responsibilities among tasks. During this work strong emphasis was layed on the design of those chains as there to guarantee that the command will be handled and may not fall off the chain.

Figures 4.5 and 4.6 both illustrate the two longest possible chains in the S-DALINAC control system: The first one is concerned with receiving, evaluating and executing device setting commands, the second one with sending status updates to the remote control. Although not directly belonging to the local control system, the remote applications are incorporated in the illustrations as they are starting resp. end points of the chain. Possible

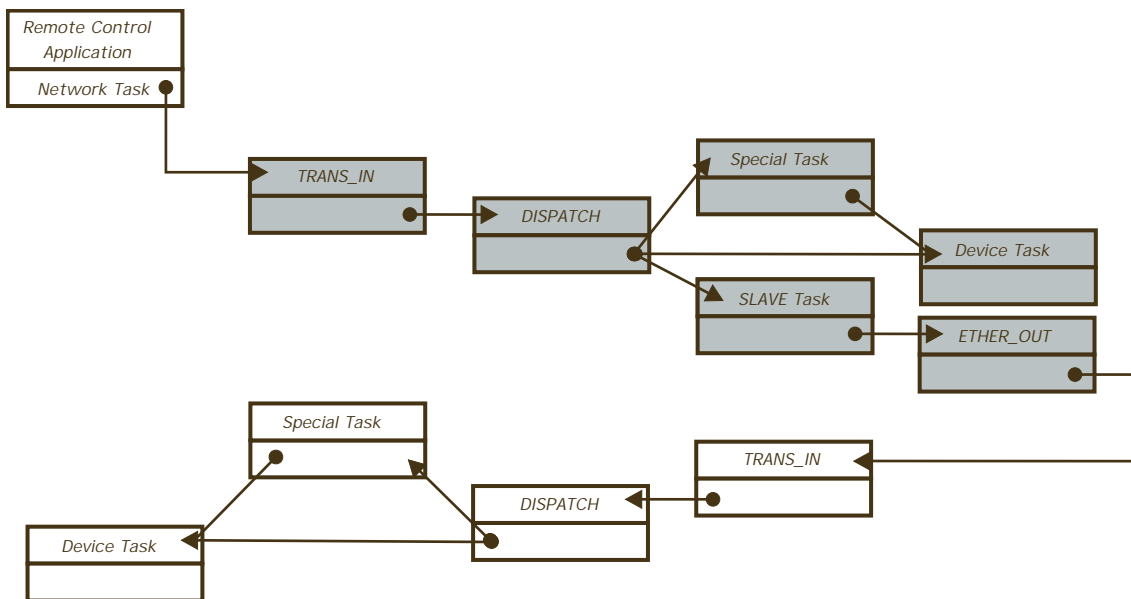


Fig. 4.5: The chain of responsibility implemented for the processing of incoming commands. A command issued by the remote control system is received by the TRANS.IN task of the master IOC (all master IOC tasks are displayed in gray). By passing the command to the dispatch task, it is either dispatched to a local device task, special task or passed along the chain through the communication IOC path to the slave IOC. Then the command follows the same chain as in the master IOC.

other sources of input are the local command interface or periodically executed commands initiated by a timer. Those chains are smaller in the number of members.

To illustrate the example of a device setting, Fig. 4.5 is discussed: Originating from the operator input at the remote control system via *ACP* or *VIEW* the command is passed to the operator interface applications' network task which adds some LCP information. Transmitted via LCP the master IOC receives the command with the *TRANS_IN* task. If the command reception has to be acknowledged, this duty is performed in this task. After removing the LCP header information the command is passed to the *DISPATCH* task that evaluates the command type. Depending on the type the command is passed to the appropriate tasks. If the command is addressed to a device that resides on the slave IOC it is passed to the *SLAVE* task, which passes the command to the *ETHER_OUT* task where the appropriate LCP headers are added and then sent to the slave IOC. The chain in the slave resembles that on the master, thus it is not further explained. If the command is addressed to a device residing on the master IOC, it is directly dispatched to the responsible device task. Should the device setting command be a more complex one, like the request to cycle a device, it is passed to the corresponding special task. The number of special tasks is not limited and during their execution other commands can be

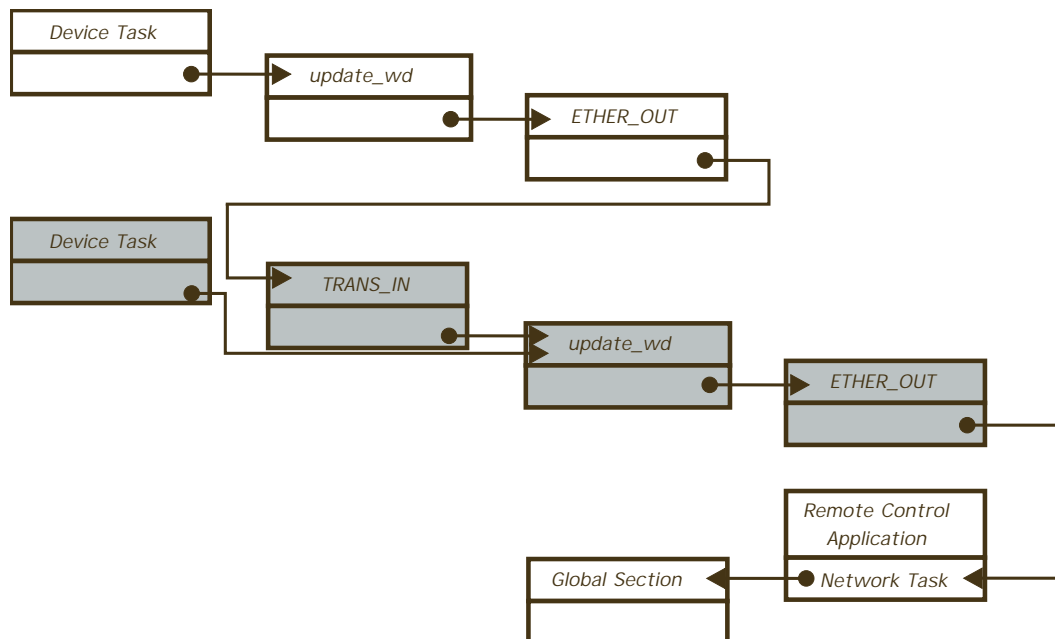


Fig. 4.6: Shown is the chain of responsibility implemented for status updates, originating from device tasks either in the slave or master IOC. The messages are passed to the *update_wd* routine, which in turn passes it to the *ETHER_OUT* that transports the message to the link partner. The master's chain is again displayed in gray. The receiver remote control network task updates the global section which represents the device database on the remote side.

handled. In the case of the cycling command the special task itself is a starting point for another chain.

Figure 4.6 shows the path for the device database update in the operator interface applications indicating a change of the S-DALINAC settings. Updates originate from device tasks, either those on the master or on the slave. Discussing the longest possible chain the update is initiated by a device task on the slave. It passes the update information to the routine concerned with sending updates (`update_wd`). This routine assembles an update message and forwards it to the `ETHER_OUT` task. After adding the LCP information it is sent to the master IOC. The slave IOC never communicates directly with the remote control system, since no update of the central device database located on the master IOC would then occur. The receiving `TRANS_IN` task is then the source of the update message in the master configuration. Due to the same task structure on master and slave, message processing follows the same path as presented for the slave configuration. On the client side the OPI network task receives the update and does not inform the remote control application directly, but updates the global section. The global section is the complementary part of the central device database on the master IOC.

4.3 Consequences for the S-DALINAC remote control system

The new functions provided by the new local control system are made accessible for the operators by adding new commands to the *ACP* operator interface. Due to changed directives for beam set-up that originated from the new cycling function, the procedure to restore an S-DALINAC setting was altered, too. For a complete reference on the new set of *ACP* commands implemented see App. H.

4.3.1 Grouping of magnets

A group is formed by specifying the names of the magnets via the *ACP* command-line interface, the first one serves as the so-called root device of the group. This is the only device that remains in the dial-list after the grouping. This list holds all the device names to be displayed in the knob-board's status line. None of the other grouped devices can be accessed via the knobs and in case a command is submitted via the command-line interface it is dismissed by the master IOC to ensure data integrity for the grouped devices. To decompose the group, the `ungroup` command with the name of the master device as parameter is issued. Then the devices of the former group are included in the dial-list. A corresponding `show groups` command lists all currently defined groups.

Grouping of devices is not only applicable to the FODO-lattice structure, but it can also be employed to group any devices, except for the bending magnets. Since they cannot be

seen isolated from their correction trim coils. In the case of the cluster magnets there is only one power supply, thus one device, for the up to eight bending magnets each equipped with an independent correction trim coil. There is no algorithm to define how to adjust the correction trim coils' values simultaneously with the bending magnet. The grouping feature can also be used to combine several magnets, that are to be cycled, for time reduction purposes. The motivation for the cycling of magnets and its implementation is presented in the next section.

4.3.2 Cycling of magnets

To ensure conditions in which beam line set-up can be performed in a reproducible manner, save and restore operations must lead to a reliable accelerator setting. Due to the fact that the beam transport magnets of the S-DALINAC are not equipped with Hall-probes, the magnetization cannot be stored in the device database of the master IOC. Instead the representation of the current provided by the power supply is used. However, the hysteresis effect of ferromagnetic materials inhibits that invariably the same magnetic field is reached when a certain current flows through the magnet coils.

Figure 4.7 shows a hysteresis curve. It illustrates the effect that a higher magnetization B_2 is reached if the magnet was driven into positive saturation before the final current is adjusted. Starting from negative saturation a much lower magnetization B_1 for the same

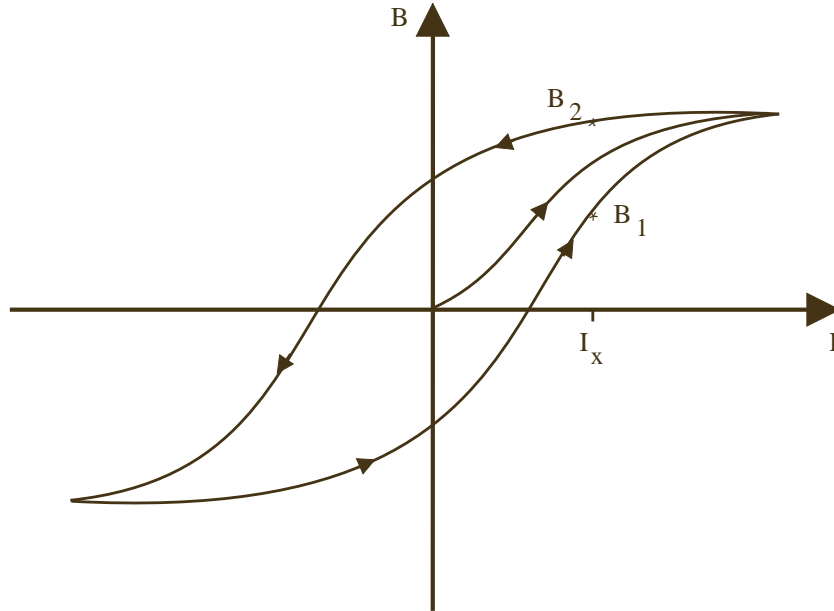


Fig. 4.7: Shown is a hysteresis curve. The magnetic field B_1 is reached at current I_x if the magnet was driven into negative saturation and B_2 if the magnet was driven into positive saturation before.

final current value I_x is reached.

If a setting is saved, the path how the final current was reached cannot be reconstructed, except if a method is defined for setting a current always with the same procedure. The remedy found is the cycling of magnets. This ensures that the path on the hysteresis curve is always the same, guaranteeing the identical magnetization. It must be pointed out that the cycling is initiated through the remote control system, but the actual cycling processes are performed in a standalone task in the master IOC.

The cycling process must be defined differently for the four types of magnet in the S-DALINAC beam line as there are:

- Unipolar power supplies for quadrupoles
- Unipolar power supplies for dipoles without any correction trim coils
- Bipolar power supplies for dipoles without any correction trim coils
- Unipolar power supplies for dipole cluster magnets with bipolar power supplies for their correction trim coils
- Bipolar power supplies for correction trim coils

Quadrupoles. The quadrupoles are cycled twice. They are ramped up in steps of 10% from zero current to their defined maximum. Figure 4.8 shows the measured hysteresis curve of a Danfysik quadrupole [36]. Then the final value is set in one step. For some of the quadrupoles the defined maximum is much smaller than the actual maximum current of the power supply because beam transport calculations showed that these maximal magnetic fields are not needed to transport the beam effectively through the S-DALINAC. The number of cycling passes for all types of power supplies was determined through visual inspection of the beam position on a view screen.

Dipoles without correction trim coils. For the definition of this specific cycling process it is not of interest if it is a bi- or unipolar power supply, because the order of negative current is so small that is irrelevant if the magnet is ramped up from its minimum current value or zero current. The cycling process is the same as for the quadrupoles except that the number of cycle passes is reduced to one.

Dipole cluster magnets with correction trim coils. This is a very special combination of two separate power supplies that drive coils in the same magnet. The magnets of the dipole clusters should all generate the same magnetic field so there is only one power supply. But due to manufacturers tolerances the magnet coils are not identical and thus not producing the same magnetic field. Therefore each magnet is equipped with correction trim coils. The power supplies for those trims are bipolar. The definition for the cycling procedure presented below is the outcome of extensive tests with the electron beam and measurements of the magnetic field in the central magnet of the first 180° bow of the first recirculation F0BM01/F0TM02:

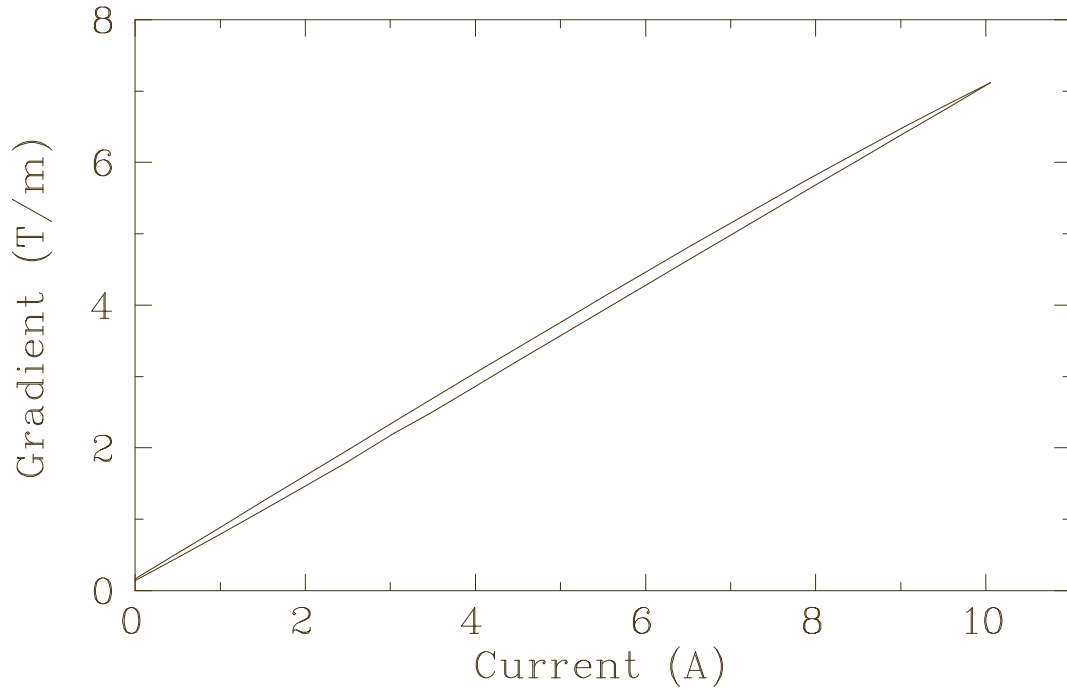


Fig. 4.8: Measured hysteresis of a Danfysik quadrupole after stepping the current from 0 to 10 A three times. Shown is the gradient of the magnetic field versus the current. The remanence is 0.154 T/m.

- First, all corresponding correction trim coils are set to their minimum value, generating a small negative magnetization. These trim settings remain constant during the cycling of the power supply of the bending magnets.
- The power supply of the bending magnets is cycled by ramping the current from zero to the maximum value in steps of 10% and then to zero again.
- After that the final current of the dipole power supply is set in one step.
- Now all corresponding trim power supplies are cycled simultaneously in the following way:
Four times a rapid five time switching between the minimum and maximum value is carried out. After each of the four blocks, the correction trim coils are set to the final value. At the very end the set value is approached from the minimum.

When the bending magnet was cycled and afterwards the correction trim coils are employed to guide the electron beam, no assumption about the actual magnetization can be made, as a small local hysteresis curve is established as shown Fig. 4.9. To gain this measurements the correction trim coil F0TM02 was cycled three times beginning at the minimum value.

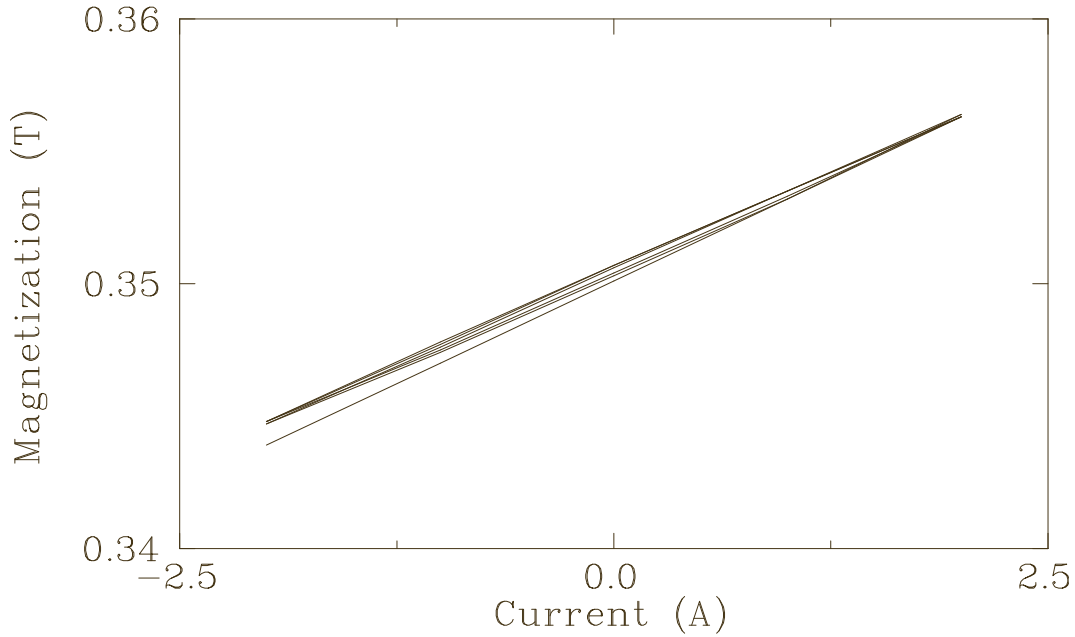


Fig. 4.9: The power supply of the correction trim coil F0TM02 was cycled three times independently from the power supply of F0BM01. After the third cycling pass the hysteresis does not change any more.

Correction trim coils. The separate cycling of a single correction trim coil, without changing the dipole setting itself, follows the same procedure as introduced above. Four times a rapid five time switching between the minimum and maximum value is carried out with a setting to the set value, before the final setting of the set value is approached from the minimum.

After each step of any cycling procedure, the current has to be hold for a certain time, so that **eddy currents** dissipate. The time to hold depends on the magnet type and is listed in the table at the end of this section. As addressed above, the operators start a cycling process and then it is carried out by the local control system. During the cycling procedure all commands addressed to the magnets, that are cycled, are dismissed by the local control system. A semaphore mechanism guards these devices during the cycling process. It is also possible to initiate a simultaneous cycling of all dipoles and quadrupoles. Then the remote control system process *ACP* is hibernating and no knob commands are evaluated, avoiding any input concerning magnet settings.

The master IOC informs the remote *ACP* when the cycling process has terminated. This is not simultaneously done with the last setting, but after a specified time to wait (also listed in Tab. 4.4), in order to assure the subside of the eddy currents.

Tab. 4.4: Idle times during the cycle process. The idle times between a single step and after reaching the set point value are listed.

Magnet type	Inter step holding time	Holding time after set point
	[s]	[s]
Quadrupoles	3	30
Correction trim coils	2	2
Dipoles	14	140

4.3.3 Influence onto saving and restoring of settings

Following the argumentation presented in the above section, the restoration of the same magnetic field in a magnet as it was present at the time when an accelerator setting was saved, requires special procedures. The reconstruction of the same current provided by the power supplies is not sufficient to guarantee the same beam transport properties. The same magnetic field has to be reproduced. Thus the saving as well as the restoration of an S-DALINAC setting is a multistage process.

Save. Before saving it must be guaranteed that all devices that have to be cycled are actually cycled. After each cycling pass, the beam position has to be controlled with the aid of the view screens or beam position monitors. When the position does not vary any longer, a reproducible setting for a single device is reached. If there are unchecked devices, the operators are informed when they initiate the saving process. The only information the local control system can pass to the remote control system is a flag indicating if a device that is to cycle, has been touched after being cycled. Only the operators can decide if the cycling process led to a reproducible result, otherwise the previous steps must be repeated. As long as no feedback loops from reliable beam position monitors are installed in the local control system, this information must be provided by the operators by visual inspection.

Restore. When restoring an archived setting, the operators are informed if any device was not cycled before the saving of the S-DALINAC beam transport settings onto hard disk. Due to the cycling of the magnets a special, multistage procedure for restoration of a setting is performed. All devices, that are to be cycled, are cycled to their minimum values. Then all, except for the correction trim coils' values are restored. To avoid an interference between the setting of the bending magnets and the trims, idle time is demanded to ensure that the requested magnetic field in the bending dipoles is established. Afterwards the correction trim coils are set to their final value. This procedure guarantees the reaching of the set-point for the magnetic field which was confirmed experimentally.

Chapter 5

Operational experience

Before the new S-DALINAC local control system was employed for its designated use, a commissioning phase was necessary. This commissioning was the initial, staged running of the whole system, beginning with hardware checks, proceeding through the software control and ending with tests for beam production. Parallel to the commissioning of the new local control system, the operators changed their procedures for beam preparation and set-up. Directives that strongly rely on the cycling function provided by the local control system were formulated.

The discussion of the operational experience is organized in two sections. The first section covers the commissioning phase where the accelerator was not yet providing beam for experiments. The second section summarizes the experiences gathered during the normal beam delivery operation.

5.1 Implementation and testing phase

Two different types for testing the system had to be considered. First the correct working of the code, second the functioning of the complete S-DALINAC beam line control system, with its remote and local control systems as well as the beam transport and diagnostics elements, was tested. The following discussion will be limited to the first type, and Chap. 5.1.2 describes in detail the commissioning phase.

5.1.1 Code implementation and testing

The implementation split up into two phases, one could be conducted off site the accelerator, and the other required access to the whole accelerator system with the remote control system and all devices. During the first implementation phase the more or less static parts of the application, listed in Tab. 5.1, were tested. The meaning of the term static resides on the fact that the influence of concurrency between the different tasks is almost eliminated. Only during the startup phase, when the tasks are created, the initialization parts run concurrently but do not interact with each other. Thus they can be seen as isolated tasks.

Tab. 5.1: Elements of the application tested off site the accelerator.

Task spawning	Includes setting up the message queues.
Device database	Creation and correct set-up of the device database with the syntax and semantic checks of the input file.
Communication with the device controllers	Correct VMEbus mapping and the access to the device controllers, not yet the devices themselves.
Driver installation	Expansion of the I/O system with the various device drivers.
Device adding	Incorporation of all accelerator devices into the application.

Only requiring access to an Alpha SBC, the first two items could be tested without need for additional hardware. The features implemented at this stage solely make use of routines provided by the VxWorks operating system.

The communication with the device controllers was first tested by passing the requests to the VMEbus without using any I/O system functions. As this is the transition between soft- and hardware, not only the software components must be tested, like mapping the correct VMEbus space into the memory of the Alpha processor, but also the correct signal processing on the VME-Q-bus converter as well as on the device controller boards. Starting from the work carried out by [32] and with help of [37] these boards were developed, modified and tested. This included the test of the so-called Programmable Array Logic chips (PALs) which implement the interface between soft- and hardware. These custom configurable Integrated Circuits (ICs) are programmed by burning logic equations into them. Through those equations the selection of the controller cards is managed, and the correct VMEbus signals resp. those for the device controllers are generated. For further information see Appendix E.

Since the former local control system hardware broke down before the work presented here had been finished, very stringent time constraints were layed on the completion of the system. Therefore the final tests on compatibility were performed on the production system, see Tab. 5.2. An incompatibility between the controllers developed on site and the commercial boards used for serial communication and data acquisition was detected at the beginning of the on-site testing phase. But this problem was fixed by using a master-slave Input/Output controller arrangement, which was now no longer an option but a mandatory requirement to operate the S-DALINAC. Having designed an open, easily expandable system, the incorporation of the slave IOC into the local control system system was possible without further code extensions.

Tab. 5.2: Tests performed on site the accelerator (production system).

LCP implementation	Tests of the communication of the master IOC with both types of remote input interfaces, the OPIs (<i>ACP</i> and <i>VIEW</i>), and the knob-server as well as the master IOC and the slave IOC.
QM80 operation	Test of the QM80 current measurement instrument.
Serial communication	Test of the serial communication with the magnets power supplies.

Looking onto the LCP encoding, extensive tests had to be performed to guarantee the robustness of the protocol implementation. Not only the right acknowledge mechanisms had to be checked but also the behavior when packet losses occur. This is only the case when the traffic on the Ethernet is too high for the NIC to filter each packet. Since such phases of high traffic are unlikely to occur, special tools for traffic generation were utilized and packet losses were also simulated in software to cover all possible LCP failures.

Measuring the beam current at the presently eight Faraday cups is an essential need for carrying out experiments at the S-DALINAC. Only one measurement instrument exists which is connected to the a multiplexer located in the accelerator hall. This device could neither be transported to the test site laboratory nor could it be simulated.

The power supplies located in the experimental hall are controlled over serial line interfaces. As the power supply local controllers are more or less, depending on the type, intelligent controllers their behavior could only rudimentarily reproduced in the test site laboratory. All power supplies controller that are addressed via serial lines are capable of either echoing the command to signal the receipt of the command; some of them even can send status messages upon request.

5.1.2 Commissioning of the new S-DALINAC local control system

After the breakdown of the former local control system hardware, the accelerator was not fully operational for a longer period of time. Thus the commissioning of the new local control system offered an additional chance to completely check all of the S-DALINAC beam transport and diagnostics units. This occasion resulted from the need to check the device repository, the source for the device database, for correctness and completeness. Errors in the database could have lead to a mismatch between the device the operator wants to influence and the device actual set to another value. But this source only includes the information about the device characteristics and for the magnet power supplies, that

are addressed through the parallel interfaces. It stores the information of the crate and slot number of the interface cards. There is neither a deductible correlation of crate and slot with the power supplies position in the racks nor with the actual magnet in the beam line. Therefore the chain of assignments was reviewed and revised. In certain cases this revision proved essential since a few correlations were wrong. For example some horizontal and vertical steerer pairs were permuted. Defect power supplies were detected as well as missing cabling or defective magnets.

The behavior of the correction coils is now consistent throughout the complete beam line, for positive values the magnetization of the cluster dipole is enhanced and for negative values reduced. This correction could not be carried out in software but had demanded the reversion of polarity of the coils.

First experiences with the cycling procedures were gained and some of the procedures have been revised after testing them during beam production. The number of minimum–maximum cycles for the cycling of a correction trim coil was figured out experimentally before encoding it in the local control system. To reduce the hysteresis effect for the quadrupoles the maximum currents for many of them were adjusted to lower values. Otherwise it would be highly recommended to cycle that device if a too high value was set by accident, before proceeding with beam set-up. Small changes while setting up the beam only requires the cycling procedure when an acceptable beam setting is found.

The topics discussed above are rules deduced from the daily operation of the S-DALINAC. These rules are not under control of the software and may not be implemented in software since they are collected in directives and beam set-up procedures by the operators [41].

5.2 Experience in daily use

During normal operation the software errors not yet detected in the development phase were removed and some software parameters were adjusted. Except for the introduction of the new type of static composite devices, no additional extensions to the original design concept had to be carried out.

Having addressed the procedural proceeding of beam production in the above section, daily experience demonstrated that the reproduction of the same beam parameters is not yet possible since the influences of the rf acceleration system (gun and cavities) cannot be neglected. Slight adjustments to the rf settings frequently entail adjustments of the beam transport devices. Nevertheless, restoring a setting, saved after having cycled all devices, reduces eminently the time to deliver the beam to the experimental site. From the operators' view the S-DALINAC beam transport system is no longer an accumulation of isolated magnets but a system that now consists of distinct sections, such as injection or first and second re-circulation. In those sections the restored settings of the magnets do not have to be adjusted. Required corrections can be made either with the rf or the magnet at the entrance of a section.

Chapter 6

Perspectives

The new local control system for the S-DALINAC beam transport system developed during the presented thesis came into daily operation six month ago. The electron beam was transported successfully to the QCLAM spectrometer. With the set-up for the 180° scattering facility [38] the ^{12}C spectra shown in Fig. 6.1 was obtained [42].

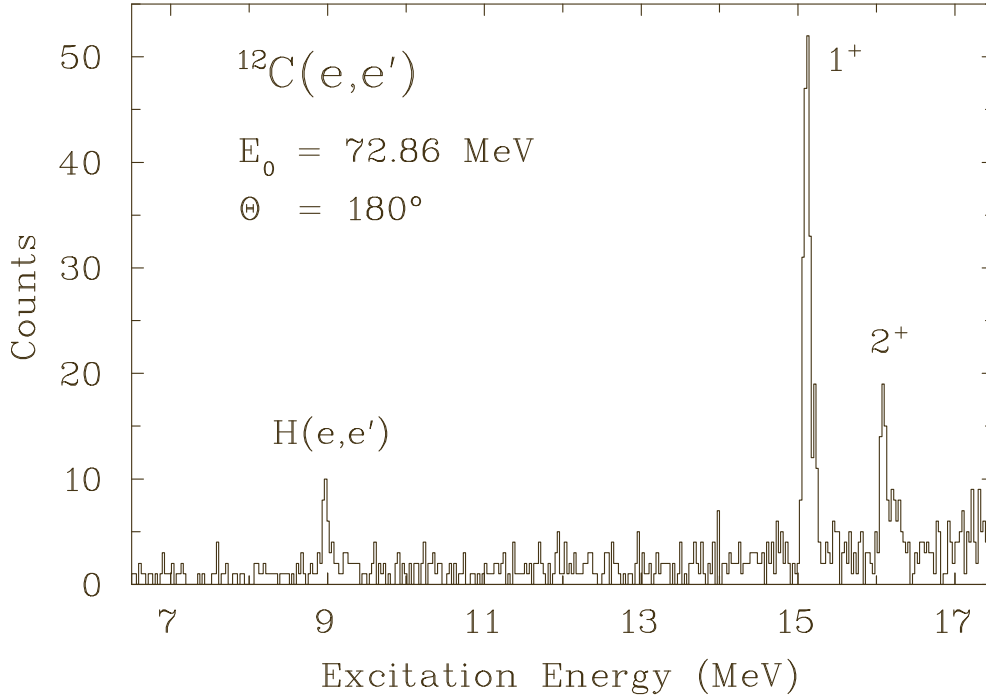


Fig. 6.1: Inelastic electron scattering spectrum of ^{12}C taken at $\theta = 180^\circ$ and $E_0 = 72.86 \text{ MeV}$ in the excitation energy range $E_x = 6.5 - 17.5 \text{ MeV}$. The line at 9 MeV originates from the elastic scattering off a ^1H contamination. The famous M1 excitation at 15.109 MeV dominates the spectrum.

The object-oriented design approach proved prospective, e.g. the actual installation of a second Input/Output controller in the system was carried out in less than a week. The implementation in such a short period of time clearly demonstrates the superior

design of an open system. Due to the specification of well defined interfaces, to the operator interfaces as well as to the inner data structure, future enhancements are easy to incorporate without a change of the overall design.

Extensions of the beam transport and diagnostics system, like the beam line to the recently installed experimental set-up for the investigation of the polarizability of the nucleon [17] or the Lintott beam line [15], are either already prepared or just require minor extensions to the device tasks. Limitations in the number of devices to be controlled are virtually non-existent. Since the VMEbus as an industrial standard is supported by a huge number of hardware manufacturers the new S-DALINAC beam transport control system can profit from a large pool of commercially available interfaces. The operating system VxWorks is also available for many hardware platforms, thus the exchange of the AXPvme system controller with another type of SBC is possible without reimplementing of the control system software. Adaptation to the underlying hardware is provided by individually tailored Board Support Packages, which are supplied by the SBC manufacturer.

Set-point control for the magnet power supplies is provided by the ADC interfaces. The current design of the controllers does not allow a read-back function. That is, the current settings cannot be read from the device interfaces but must be derived from the database values. To have the QM80, view screens, and monitors reflect the actual settings, a revision of the on site developed device controller cards must be carried out. Alternatively a complete redesign of the target control towards a CAN-based bus system [39] is an option. This field bus is a very popular one in the European market, since it excels in robustness and scalability. Many accelerator sites already employ that field bus. In contrast to the existing system, where each view screen is driven by a private control line, CAN implements a bus topology. This will drastically reduce the amount of cabling needed.

The above outlined improvements will conclude the actual local control system. Further emphasis should be put on the remote control applications. While the new local control system is no longer limited to DECnet-like protocols, the legacy knob-server still enforces them in the overall S-DALINAC control system. Beyond that, the hardware used for the Ethernet interface of the knob-server is not capable to support sustained data rates over 300 kbyte/s. Receiving higher byte rates does lead to severe hardware failures on the knob-server module. A redesign of the knob-server should provide a solution to these drawbacks. The implementation of the open-system design for the local control system puts no constraints on the selected communication protocol. By choosing the IP protocol suite the implementation of Web- or Java-based operator interfaces is practicable. This type of operator interface is hardware independent, it may run on any available workstation in the network. The concept that the operator interface has an exact copy of the device database can be omitted. The master IOC then would tell the clients about all devices administered when a connection is established. Another drawback that will be eliminated when the knob-server is replaced, is the demand that its boot-host is part of the OpenVMS cluster. Without this, the complete S-DALINAC beam-transport control system could form an autonomous system with its private network segment. Two different

versions of this separation may be realized:

- Private network segment with limited interconnection to the laboratory LAN. This version enables an easy transport of data between the hosts of the accelerator segment and the hosts and peripheral devices located in the other segments for further data processing. The development host for the local control system could reside in its current segment. Influences from outside may still interfere with the accelerator operation.
- Isolated private network segment with no interconnection to the laboratory LAN. This version eliminates any influences originating from outside the accelerator segment. As a drawback all necessary peripherals such as printer, data storage, backup and development host have to be provided in this isolated network. The synergy effect of sharing peripheral devices is no longer feasible.

The work carried out in this thesis provides a fundamental new basis for the complete accelerator control system. The change from hardware dependent towards hardware independent implementation of the new local control system supported by the object-oriented approach, proved so successful so that now the remote operator interfaces should also be revised. The then completely modernized S-DALINAC control system will be competitive to other modern accelerator control systems.

Appendix A

Comparison of accelerator control systems

Accelerators, that are explicitly built for experimental purposes, must have a control system designed to meet the special requirements and reflect the unique purpose of the machine. Those requirements may immensely differ from accelerator to accelerator thus the comparison of the control system used at the S-DALINAC with other accelerators' control systems will be limited to design topics only. Even those sites using EPICS, the **Experimental Physics and Industrial Control System** [6], build a private system from the EPICS toolkit. This is a set of software tools and applications providing a framework for developing a control system application. Various other accelerator sites have developed their own control system. But all more modern systems share the common approach for a client-server model. The software architecture strongly relies on an object-oriented design and implementation methods. The connection between the clients and servers is provided by an IP-based LAN. The protocols employed, are TCP, UDP and RPC; in some rare case private protocol definitions, like the LCP, are employed.

The client applications are mostly graphical user interfaces (GUIs), some of them provide additional support of digital rotary knobs. The experiences at the S-DALINAC with GUI-based OPIs are diverse:

- The *VIEW* application for the view screen, Faraday cup and camera control is a graphical one with a trackball, a mouse like device, as input device. The *VIEW* provides the operators with the complete status of the view screens and monitor assignment at a glance.
- The first application developed for the beam-transport remote control system, *MCP*, was GUI-based. The actual layout of the beam-transport was displayed on screen and the operators could manipulate a device after selecting it. Due to the finite size of the computer monitor and zooming capabilities of the application, this handling was judged not appropriate by the operators. Thus a new type of beam-transport application, the *ACP* equipped with a command line interface, was implemented.

Compared to other accelerators, the amount of devices to be controlled and monitored at the S-DALINAC is relatively small and the operating staff does not fluctuate frequently, so that command-line-based interfaces, that are supported by the knobs, are sufficient. The syntax of the commands and the naming convention for the devices is intuitive so that a support by an graphical user interface is not necessary.

- The OPI for the rf system, *HF*, provides also a command-line-based interface, but here the interface is combined with the display of different masks on the terminal screen. This combination proved so successful, that the re-implementation [7] of this OPI used the same toolkit for the display of the data.

Many other control systems split the local control server in two layers. This builds up a multi layer server structure. One layer is concerned with administering the device database with a commercial available database system and processes data gained by the other layer that also controls the devices. This second layer works independent from the process layer, it solely transforms the setting and reading commands into the code required for the device controllers. Those control systems use field busses for interaction with their accelerator devices.

With this kind of implementation, the process layer does not need to be extended when new or additional devices are added to the field bus layer. At startup the control system reads the current configuration of devices for each field bus Input/Output controller. Each field bus system dynamically builds its own internal database which is cross checked with the database system in the process layer.

While this approach allows the implementation of closed-loop algorithms that perform online beam transport calculations and optimize independently the beam set-up without operator interaction, it adds more complexity to the overall system. At the S-DALINAC this is not required because:

- the number of devices to be controlled is small.
- the device database is already mirrored to the operator interface applications so that online beam transport calculations can be performed. For the S-DALINAC based on the work of [40] an online beam calculation program [18] was developed. This program does not influence the beam properties directly. It solely gives the operators information at hand with devices to chose to optimize the beam.
- a commercial database system requires an additional powerful host system and does not work with the knob-server.

Appendix B

S-DALINAC control system

This appendix should serve as a very brief introduction to the S-DALINAC control system for an programmer, who is already used to the terminology and design processes for real-time applications but must adopt them to the S-DALINAC control applications. This overview will also enclose the rf-control system. Both control systems (rf and beam transport) strongly rely on the LCP as interconnecting network protocol and the knob-server as an instrument of the operator interfaces. Hard- and software components are discussed in separate sections. Concerning the network link, the general properties of the local network will be discussed in the hardware section, whereas the S-DALINAC control system private protocol, the Linac Control Protocol, is presented in the software section. This separation helps structuring the discussion of the complex S-DALINAC control system. The following figure illustrates the overall system. The shadowed parts mark the network protocols the hosts are capable to interpret. The dotted line boxes on the local control system side show the attached controllers in the VMEbus crates. In the other boxes the host-name, processor type, type of the physical network interface and the operating system are displayed.

B.1 Hardware overview

The laboratory Local Area Network, shared by both the remote and local control, is an Ethernet based network. The cabling media is ThinWire (10BASE-T) that is structured in a bus topology. All local control workstations, except the knob-server, are connected with BNC connectors to the network. The knob-server's network interface is an AUI connector, the same type as for the other VMEbus based systems. To connect the system to the network either a transceiver, a device used to transform the electrical signals on the coax cable into signals transported over an 8-conductor special cable (transceiver cable), or a DELNI must be employed. This is a multiport repeater for AUI, providing up to eight AUI ports and one BNC connector. Therefore one of those DELNIs is used in the klystron room, where the local control systems are located. This also enables the deviation from the bus topology enforced on a 10BASE-T Ethernet to the star like topology for the VMEbus systems illustrated in Fig. B.1. The AUI cable can be up to 50 m long but connects only one station to the Ethernet; the cable is terminated in the unit.

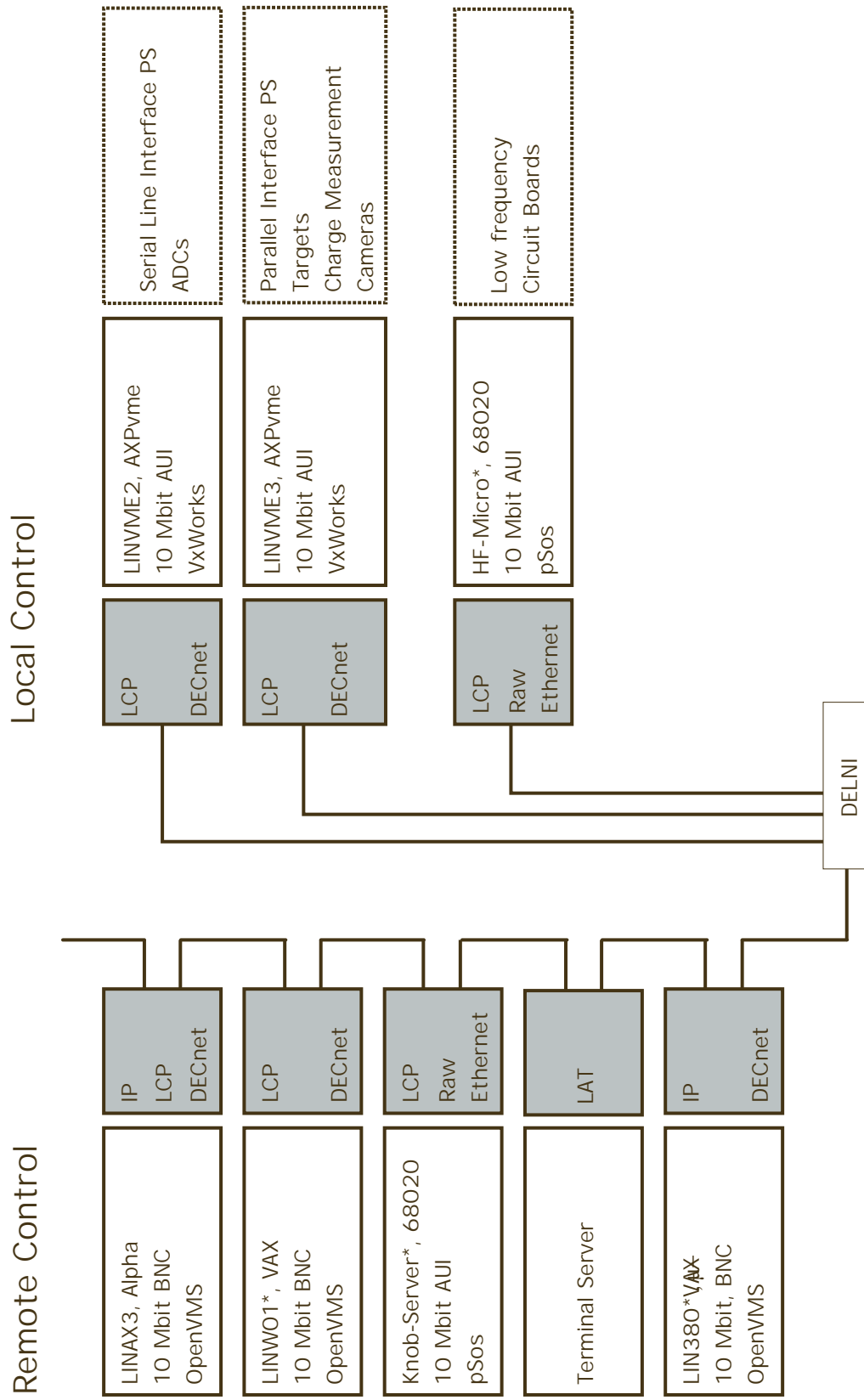


Fig. B.1: Abstract view of the complete S-DALINAC control system. The shadowed parts show the network protocols the host has implemented. The hosts indicated with an asterisk are not capable to use the full Ethernet bandwidth.

The specification for the 10Base-T cabling specifies 30 stations on and up to 185 m long RG-58 type cable. The $50\ \Omega$ resistors used for proper termination already count as two stations, leading a total of 28 stations to be interconnected. Each network interface has to be connected with an T-connector assuring the compliance with the bus topology. Each station must at least be 30 cm apart and any other cable segmentation counts as one station.

The maximum transfer rate is 10 Mbit/s, but some of the network interface electronics used in the control system are not capable of transceiving higher rates than 300 kbyte/s, those computer systems are marked with an asterisk. They may fail, even in hardware, if packets addressed to them are incoming at a higher rate or they may loose packets while booting their system image as explained further down.

Table B.1 assorts the hardware employed in the remote control systems. Next to the name of the computer, the operating system and version number, the processor type, the hardware addresses in their canonical form (the worldwide unique hardware address and the private DECnet address) and the IP address are listed.

The micro processor board serving as knob-server does not have a specific name in the network, but to identify it in the table it is listed under this name. It should be pointed out that despite working under the same operating system there are distinct differences between a computer system based on a VAX or an Alpha processor. The width of the internal bus is 32 resp. 64 bit. Thus code compiled under a VAX architecture is not executable on an Alpha architecture and vice versa. The executable for the Motorola 68000 architecture has to be compiled on a VAX host using the 68000 cross compiler. The knob-server consists of two VMEbus modules and one break-out module providing the parallel interface for the knob-boards employed in the control room. One of the two boards is the processor board equipped with four serial lines, of which one is used as

Tab. B.1: List of the computer systems used in the S-DALINAC remote control systems. The MAC address in the first line gives the DECnet address and the second line the manufacturers Ethernet address. A dash indicates that the system has not installed IP.

Name	Processor type	MAC addresses	IP address
LINAX3	Alpha	AA-00-04-00-F4-05 08-00-2B-BC-A2-63	130.83.133.31
LINW01*	VAX	AA-00-04-00-F4-05 08-00-2B-0B-0C-2C	—
LIN380*	VAX	AA-00-04-00-A6-06 08-00-2B-11-BD-11	130.83.133.3
Knob-server*	Motorola 68020	AA-00-04-AA-AA-3F —	—

console line and the others are used to connect up to three portable single row knob-boards. The other board is the Ethernet interface. The VMEbus backplane is not a standard 21 slot one, but the system does not depend on this specific backplane. The underlying bus systems for the workstations have no relevance in this discussion.

The host LIN380 is listed in the above table even though it is not used for local or remote control applications. It can serve as a development host but more important is the boot-loading host for all micro processor boards employed at the S-DALINAC and its experimental facilities – including the FEL micro processor and the one used for data acquisition with the QCLAM multi-wire drift chambers. This system is slow enough to send packets over Ethernet for reliable downloading of the application image into the micro processor boards. The firmware of those systems, that is burned into EPROMs, has not implemented any higher OSI protocol layers, so that any possible packet loss would lead to a failure of system initialization.

The computer system LIN380 will not be included in the next table, giving an overview of the hardware used for the local control system. Similar to the case of the knob-server micro computer the rf control system has not assigned a qualifying name, so that the term HF-micro is used. Both the LINVME2 and LINVME3 are hosted by a standard VMEbus crate (the slot #21 cannot be used since the backplane termination occupies this slot), but due to the higher electric power consumption of the Alpha processor, the SBC needs the power supplied from two VMEbus slots. For this purpose a break-out module is attached to the backside of the backplane, routing more power to the first slot. So if ever the

Tab. B.2: Number and type of the VMEbus modules in the different VMEbus crates of the local control systems followed by a short description of their purpose.

System controller	VMEbus module	Purpose
LINVME3	5 Q-bus-VMEbus converters	Converts the VMEbus signals into Q-bus like signals interpretable by the parallel interface controllers, the target and camera/QM80 controller.
LINVME2	1 serial line interface board	Provides up to eight RS 485 or up to 16 RS 232 serial lines, or a mixture of both due to the modularity of the board.
	1 14-bit 64 channel ADC board	Interface for various beam monitoring devices or monitoring the rf.
	4 12-bit 16 channel ADC boards	Provides set-point control for the power supplies.
HF-micro*	14 low frequency boards	Each board controls one of the accelerator cavities and the chopper/prebuncher.

crate must be replaced or used with another SBC type, this break-out module must be removed. Otherwise the SBC will be seriously damaged or even a total loss of the system may occur.

The crate hosting the rf control system has a customized backplane, not exchangeable with any other crate employed in the control system, neither the remote nor the local. The upper part of the backplane, or in other terms the P1 connectors, are standard VMEbus, the lower part, namely the P2 connectors, are adapted to the needs originating from the operation of the low frequency boards. Table B.2 lists the different controllers and bus modules necessary for the S-DALINAC control system. They are grouped in the same way as they are distributed over the VMEbus crates.

B.2 Software overview

The software overview follows the same structure as the hardware presentation. The LCP as interconnecting network protocol is discussed first and afterwards the remote and local control systems. Both systems are presented together with the operating systems on which they execute. The presentation of the operating systems is crucial as the applications are strongly interwoven with them.

The LCP is implemented above the IEEE 802.3 Ethernet protocol and extends the payload of the packet with an additional header, see App. G. Depending on the ability of the hardware and software the connectionless or connection-oriented version of the LCP is implemented in the applications. The micro-processor based systems implement the connectionless protocol derivative because the overhead to maintain a session is too large. Table B.3 itemizes the different applications on the hosts of the remote control. Deriving from the

Tab. B.3: Software components building up the remote control system of the S-DALINAC. The applications are listed together with type of LCP they have implemented - the abbreviation CL means connectionless and CO connection-oriented. The network partners are also listed.

Host	Application	LCP type	Network partner	Subnet protocol-ID
LINAX3	<i>ACP</i>	CO	LINVME3	08-00-2B-60-12
		CL	Knob-server	08-00-2B-60-10
	<i>HF</i>	CL	HF-micro	08-00-2B-60-11
		CL	Knob-server	08-00-2B-60-10
LINW01	<i>VIEW</i> (old)	CO	LINVME3	08-00-2B-60-12
	<i>VIEW</i> (new)	CO	LINVME3	08-00-2B-60-16
LINAX5	<i>CCP</i>	CL	EMI	08-00-2B-60-15

fact that more than one OpenVMS application is running on one of the hosts, it is necessary to specify different private subprotocol headers to uniquely identify the channel opened on the Ethernet interface. An open channel is a prerequisite to access directly the DECnet-protocol layer in the OpenVMS operating system. For incoming packets the NIC can assign the frame to one channel only. Each application must have a unique subnet protocol-ID when running on the same host, so that the NIC can distribute the frames among them. Currently the *ACP* and *VIEW* are running on different hosts using the same subnet protocol-ID, but with the implementation of the new *VIEW* [11] they may run on the same host with different subnet protocol-IDs. All Alpha/VAX hosts are running under OpenVMS V6.2 and the knob-server, just as the HF-micro, under the real-time operating system pSOS.

To state that the protocol-ID for the FEL experimental control system (*CCP*), hosted by the micro controller EMI, is not available to S-DALINAC control system programmers, it is listed in the table even though it is not part of the accelerator control system. For further information on the *CCP* see ref. [43].

The overall structure of the beam transport and diagnostic applications *ACP* and *VIEW* (new) is the same, they differ only in the user interface. The *ACP* depends very much on the knob boards as input devices, whereas the new *VIEW* should provide a comfortable graphical user interface. But both are depending on the global section as source of the information of the status of the accelerator.

Appendix C

System installation and upgrade

C.1 AXPvme Single Board Computer

When installing the AXPvme SBC in a new crate it is very important to remember that this SBC needs the power supplied by two VMEbus slots. A so-called **breakout module** must be inserted on the backside of the backplane at the P2 connector. For more information refer to [44]. With this **breakout module** installed, no single slot module must be inserted in either slot #0 or slot #1, otherwise the module gets severely damaged.

As there is no hard-disk available in this system, the AXPvme SBC must be booted via the network using the **bootp** and **tftp** protocols. Downloading and booting are both two step processes:

1. Initially the SBC requests its Internet address, the address of its bootp server and the boot file name via the bootp protocol. The responding bootp server must be set up according the syntax required for the **bootptab** file to provide this information. On an OpenVMS host these services are provided through **UXC** and on a Digital Unix host through the **joind** daemon.
2. By using tftp the kernel is downloaded into the SBC and the AXPvme SBC console program starts it. This step is the last in the downloading sequence and is simultaneously the first concerning the boot process.
3. The second level boot already uses the VxWorks drivers, thus the boot is no longer controlled by the console but by the parameters delivered in the VxWorks bootline. The S-DALINAC beam transport control system bootlines for master (a) and slave (b) are:
 - (a) `tu(0,0)linux2:/userc6/users/sdalinac/vxw/bootp/VxWorksMaster.st
h=130.83.133.7 e=130.83.133.67 u=sdalinac tn=linvme3`
 - (b) `tu(0,0)linux2:/userc6/users/sdalinac/vxw/bootp/VxWorksSlave.st
h=130.83.133.7 e=130.83.133.66 u=sdalinac tn=linvme2`

Booting from Ethernet requires two different loader files:

- the secondary loader file `netload.AXPvme` and
- the actual kernel `vxWorks.st`.

Special attention should be turned to the setting of the AXPvme SBC VMEbus slave base addresses. Any VMEbus master can operate as a VMEbus slave, but the board's base addresses for VMEbus master and VMEbus slave operation must not overlap. This definition has nothing to do with the terminology of master and slave IOC; and the definition of the base addresses are private to each AXPvme SBC. Sizes and base addresses can be set in the firmware environment variables at the console prompt. Table C.1 shows the default settings for the S-DALINAC control system. Changing these default values can be done using the `set vme*` commands. For the A16 the size is a fixed 256 byte window. For the A24 and A32 one can define the size from 64 kB up to 16 MB window (A24) and 16 MB up to 128 MB window (A32). If the window boundary does not fall into a valid boundary, VxWorks adjusts it correctly.

Tab. C.1: VMEbus slave base addresses and sizes for the master and slave IOC.

VMEbus address space	Master IOC	Slave IOC
A16	0x000000	0x000000
A24	0x00FF00	0x00FF00
A32	0xFF0000	0xFF0000

By specifying the value zero for the `vme_a24_size` and `vme_a32_size`, the default values from the `$WIND_BASE/config/AXPvme/conFig.h` are used.

C.2 System upgrade

System administrators or developers have to create their personal development environment where the new applications can be built and kernel modifications can be tested. The VxWorks software is currently installed on the `LINUX2` host. The `PATH` environment variable defined in the private `.bash_profile` should be extended with the element `/usr/opt/VXW531/host/gdb/bin/osf` to make use of the development utilities VxWorks provides. To download object files to the AXPvme SBC or to write files onto disks, `rsh` access methods are used. This implies the use of a password which can be avoided by including the node name of the AXPvme SBC in the `.rhosts` file located in the programmer's login directory.

The system files located at the `/usr/opt/VXW531` directory should never be modified. Changes to the kernel should be carried out in a local copy of this directory tree. The

`shadow` command, creates such a local copy by duplicating the directory structure but only linking the system files to the source directory. Files to be modified must first be copied from the source location to the correct path in this shadow tree.

Mandatory changes to build the S-DALINAC control system kernel can be found in the following locations:

- `/userc6/users/sdalinac/vxw/target/config/AXPvme/conFig.h`
- `/userc6/users/sdalinac/vxw/target/config/all/configAll.h`
- `/userc6/users/sdalinac/vxw/target/config/all/usrConFig.c`

Using the original configuration files instead of the above mentioned would lead to a non-functioning control system.

For successful system building the host system must be informed about the environment variables for the makefile, those are given in Tab. C.2

Tab. C.2: Environment variables needed to build a VxWorks system image.

VX_VW_BASE	location of the shadow tree
VX_HSP_BASE	location of the shadow tree (same as VX_VW_BASE)
VX_HOST_TYPE	specifies the host system type; here OSF indicating an Alpha target
VX_BSP_BASE	location of the cloned tree
VX_MAKE	<code>\$VX_VW_BASE/h/make</code> , specifies the relative location of the make utility within the directory tree.

The makefiles for building the system images can be found in the directory `/userc6/users/sdalinac/vxw/target/config/AXPvme/` named `make_sdalinac`. The commands for master (a) and slave (b) are:

- (a) `make -f make_sdalinac VxWorksMaster.st`
- (b) `make -f make_sdalinac VxWorksSlave.st`

To make use of the new system images the AXPvme SBCs must be rebooted.

Appendix D

Inter task communication tools used at the S-DALINAC local control system

Having introduced inter task communication tools in Chap. 3.1, they will be presented now in greater detail and in the context of the S-DALINAC beam transport control system. Signals for task synchronization are not employed since there is no need to synchronize the tasks depending on the message the signal delivers. Messages are exchanged solely over message queues since no direct and synchronous coupling between sender and receiver tasks is desired.

During the design process of the S-DALINAC local control system many problems arose due to the parallelism of services the application has to offer. Inter task communication is needed when competitive processes have to cooperate to form the effective application. So the following issues have been addressed:

- mutual exclusion
- task synchronization
- message exchange

They do not occur permanently during program execution, but whenever a task enters a critical region, like updating a value in the device database. Not checking the system state before entering the critical region may lead to a state of deadlock or starvation [20]. A recovery from those states is not possible without either loss of data and data integrity or a system reboot.

For synchronization purposes solving the first two problems of the above list Dijkstra introduced the use of Boolean variables in [45]. Those variables, later called semaphores, can be taken or given only in a single, indivisible action. Once an operation on the semaphore has started, it is guaranteed that no other task can interrupt this action until the operation has completed.

Mutual exclusion. A race condition [20] arises when two or more tasks want to access a shared resource at one time. So a synchronization mechanism must be provided to ensure

that access to the resource is mutually exclusive. Illustrated in Fig. D.1 task **NHDev** must first signal access to the resource **device database** by acquiring the semaphore, then it can work on the database, and when leaving the critical region it has to release the semaphore and thereby signal the availability of the database. If task **SerialDev** wants to access the **device database** while **NHDev** holds it, task **SerialDev** has to wait until the semaphore is available. The available semaphore is characterized by a black flag, a semaphore taken by a task by a white one.

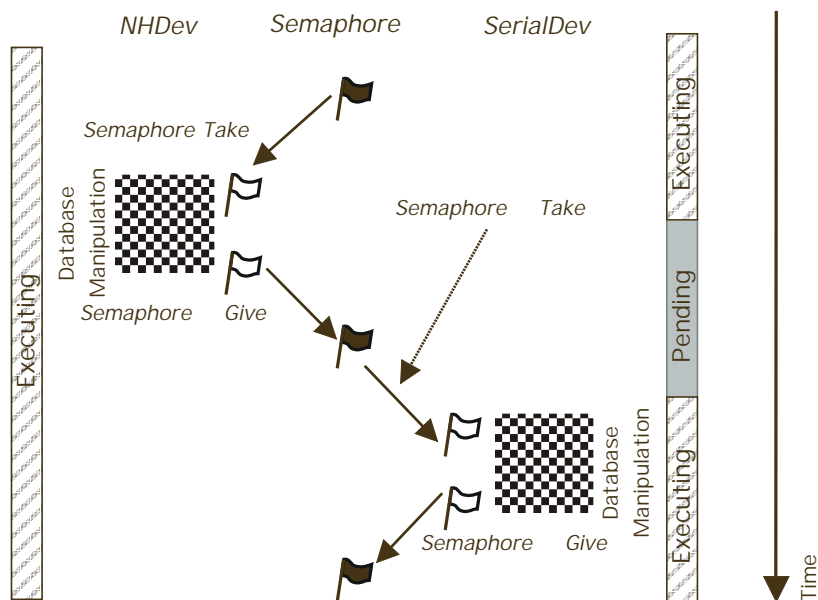


Fig. D.1: Mutual exclusion semaphore. The semaphore illustrated by the flag guards the shared resource **device database** and ensures a sequential access. Further explanation is given in the text.

Task synchronization. A similar problem to the above is the task synchronization problem. This occurs when two or more tasks have to coordinate their actions without exchanging data between each other. At the S-DALINAC control system this is used to signal the reception of an answer from a power supply that has a serial line interface. These power supplies are equipped with an intelligent controller that can bidirectionally communicate with the control system application. Figure D.2 shows such a synchronization. After a command is sent out to the power supply controller, the task **RXNL** is waiting for the answer. The end of the incoming data from the serial device is indicated by a `\cr`, which causes the Interrupt Service Routine to give the semaphore the task **RXNL** is waiting for. Then the task **RXNL** empties the read buffer and processes the response.

To distinguish between the two different operational modes of a binary semaphore, one has to take a closer look on the initial state. Employing the semaphore for mutual exclusion it

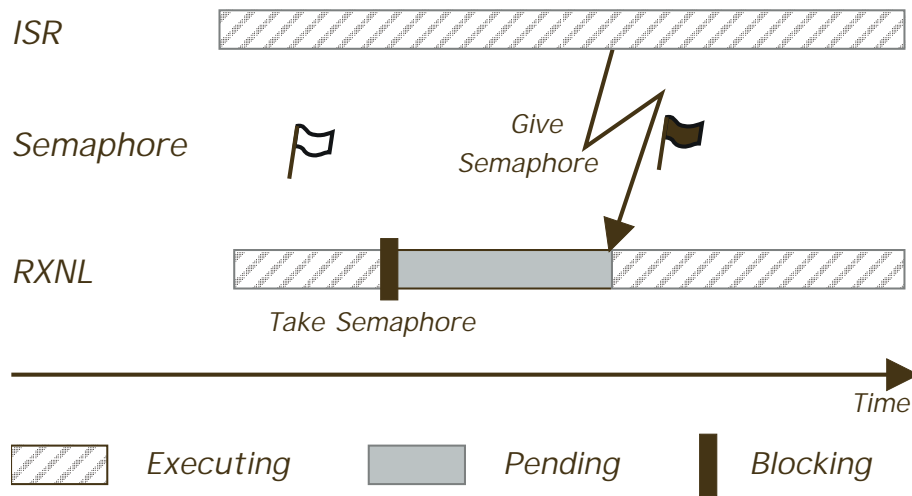


Fig. D.2: Task synchronization semaphore. The Interrupt Service Routine, ISR, gives the semaphore the task RXNL is waiting for, so that it can evaluate the ring buffer content.

is created with an initial state of **FULL**. While taking the semaphore the state changes to **EMPTY** signaling that the resource is in use. The exact opposite is the task synchronization, as used e.g. out of interrupt service routines. The semaphore is created with an initial state of **EMPTY** and to signal the state transition the semaphore becomes **FULL**.

Message exchange. Shared memory is the easiest way to pass information between tasks. All tasks have equal access to that specific data segment in memory so that there is no need for special access methods. But with the problem of multiple writers, data integrity cannot be guaranteed without a massive synchronization overhead. The device database is implemented in shared memory and protected by semaphores as discussed above.

Message queues are used throughout the system to pass commands along the various chains of responsibility. They are special devices created by one task but organized by the operation system in which tasks can write blocks of data, called messages. At creation time, the creating task defines the maximum size of the message and the maximum number of messages being stored in that queue. During operation, a task may write a message into this queue, and it can immediately resume operation since the operation system ensures that the message is stored until the intended receiver reads the data from that queue. This effectively separates the sender from the receiver. The operation system is responsible for all required synchronization and storage so that the application does not need to deal with these issues.

Appendix E

Q-bus-VMEbus converter

Introduced in Chap. 4.1 the change from the parallel Q-bus interfaces towards the VMEbus based system enforced the development of a Q-bus-VMEbus converter and new controller cards for the parallel interface crates and the QM80/camera resp. the target modules.

Table E.1 lists the relevant VMEbus signals for the communication with the controller cards. The data lines are driven from the VMEbus on the converter card where the signals are transformed into differential ones. Those signals are then passed to the controller cards, where they are converted back. The handling of the address signals is quite the same except that A00 and A01 are not transmitted. Direct access to the A00 bit is not provided by the VMEbus, it is encoded in the data strobes DS0 and DS1; A01 is not used to address any of the S-DALINAC devices.

Tab. E.1: This table lists the VMEbus signals that are used during the data transfer from the VMEbus to the device controller.

VMEbus signal	Description
A01 – A24	Address lines, used to address the Q-bus-VMEbus converter cards.
D00 – D15	Data lines, used to transfer the actual data.
DTACK	Line signaling that the data transfer can be terminated.
WRITE	Line indicating the direction: read or write.
AM0 – AM5	Address modifiers, specifying the address space (here A24).
LWORD	Line indicating the width of the data transfer.

The correct addressing is checked by the logic burned into the PAL DECODE (see Fig. E.1). There the address modifiers (specifying address space and type) and the direction of the data flow is verified. Although the address space is the A24, the bits A16 to A23 are not used for addressing purposes of a single interface. These address lines are solely used to address the Q-bus-VMEbus converter located in the VMEbus crate. To address an entirely filled parallel interface crate, the range is from 0x000 up to 0x01FF. An exception are the address bits A14 and A15. They are used to generate the ISEL0 signal, that is

mandatory to select the single interface. The PAL DTACK (see Fig. E.2) generates this signal next to the VMEbus signal DTACK*. This active low signal (indicated through the asterisk) is needed in the VMEbus data transfer cycle to signal the bus master that the receiver got the data and the bus can be freed. The higher address bits A16-A23 are solely used to address the Q-bus-VMEbus converter. They are compared with the value preset in the on board dip-switch.

```

Chip      decode      Pal16L8
/sel /uds /lds am5 am0 am1 am2 /as am3 gnd
am4 /berr /lword /we a1 /iwrite nc /isel /iread vcc

String    data24      'am5 * am4 * am3 * /am1 * am0'
Equations

isel      := as * uds * lds * sel * /lword * data24
iread     := as * uds * lds * sel * /lword * data24 * /we
iwrite    := we
berr      := as * uds * lds * sel * lword * data24

```

Fig. E.1: Content of the DECODE.PDS file. These equations are compiled with a cross compiler and then burned into the PAL.

```

Chip      dtack      Pal16R4
clk a15 /isel a14 /iwrite nc nc nc nc gnd
/oe /iwe /dtack /q3 /q2 /q1 /q0 /isel1 /isel0 vcc
Equations

q0      := /q0
        + /isel
q1      := /q1 * /q0
        + q1 * q0
        + /isel
q2      := /q2 * /q1 * /q0
        + q2 * q1
        + q2 * q0
        + /isel
q3      := /q3 * /q2 * /q1 * /q0
        + q3 * q2
        + q3 * q1
        + q3 * q0
        + /isel

dtack := isel * q3 * q2 * /q1 * /q0
        + dtack * isel

isel0 := isel * /a15 * /a14 * /iwrite
        + isel * /a15 * /a14 * iwrite * q3 * q2 * /q1 * q0
        + isel * /a15 * /a14 * iwrite * q3 * q2 * /q1 * /q0
        + isel * /a15 * /a14 * iwrite * q3 * q2 * /q1 * /q0
        + isel * /a15 * /a14 * iwrite * q3 * /q2 * q1 * q0
        + isel * /a15 * /a14 * iwrite * q3 * /q2 * q1 * /q0
        + isel * /a15 * /a14 * iwrite * q3 * /q2 * /q1 * /q0
isel1 := isel * /a15 * a14 * /iwrite
        + isel * /a15 * a14 * iwrite * q3 * q2 * /q1 * q0
        + isel * /a15 * a14 * iwrite * q3 * q2 * /q1 * /q0
        + isel * /a15 * a14 * iwrite * q3 * q2 * /q1 * /q0
        + isel * /a15 * a14 * iwrite * q3 * /q2 * q1 * q0
        + isel * /a15 * a14 * iwrite * q3 * /q2 * q1 * /q0
        + isel * /a15 * a14 * iwrite * q3 * /q2 * /q1 * /q0
iwe     := iwrite

```

Fig. E.2: Content of the DTACK.PDS file.

The description of the Q-bus – VMEbus converter is not exhaustive before the new controller card for the parallel interface cards, the view screens, the camera multiplexer and the QM80 is described. The data lines are driven from the controller card to the units without further processing. The parallel interface cards are controlled in a slightly different way than the other units. Thus the discussion splits up in two parts:

- The parallel interface cards are addressed by a single **select**-line (upper part of the backplane), enabling the interface card to read the data and control lines from the P2 connector (lower backplane part). This design implies that almost the addressing lines A03 – A14 must be decoded to this single **select**-line. To address a channel on the interface card the VMEbus address lines A02 and A03 are not used to generate the **select** signal but used to address the channel. The naming convention for those lines is on this board A1 and A2. For address decoding the PAL DEC_NETZ, see Fig. E.3 is employed, which also generates the INV* signal, that is active as long as no data is transmitted from the interface cards to the VMEbus.
- The second type of controller cards (view screens, QM80 and cameras) only encodes the address lines A04 – A06 to generate the **enable**-line. An offset of 0x180 (address lines A07, A08) must be added, to generate a signal that triggers the *3to8*-converter to generate the **enable**-line.

```

Chip      dec_netz      Pal16L8
/isel0 /ia7 /ia8 /iread /brdsel nc nc nc nc gnd
nc nc nc nc /cs3 /cs2 /cs1 /cs0 /ivn vcc
Equations

cs0 := isel0 * brdsel * /ia8 * /ia7
cs1 := isel0 * brdsel * /ia8 * ia7
cs2 := isel0 * brdsel * ia8 * /ia7
cs3 := isel0 * brdsel * ia8 * ia7
ivn := /isel0 + /brdsel + /iread

```

Fig. E.3: Content of the DEC_NETZ.PDS file.

Appendix F

Device drivers under VxWorks

One of the main functions of the operating system is to control all the computers' I/O devices and to provide a uniform interface that is easy to use and common to all devices and device controllers. This junction between operating system and user application is provided by device drivers, which form a device independent access. This concept relies on the fact that a large fraction of the I/O software is device independent and only some of it is device specific. The S-DALINAC local control system uses two types of device drivers [46]:

- **Character drivers.** They can handle I/O requests of arbitrary size and can be used to support almost every type of device.
- **Terminal drivers.** They are specialized character device drivers to deal with communication over serial lines.

F.1 Driver entry points

The uniform interface is provided by adding seven basic I/O routines to the operating system, building up the driver. In these routines all private information about how to access the device is stored. The application programmer does not need to know how to call these routines since the normal C programming or POSIX function calls, like `fopen`, `open`, or `fprintf` for device accessing can be used. The seven entry point functions, listed below in Tab. F.1, are added to the I/O system when the driver is installed. These routines are then called by the kernel when the C or POSIX programming functions are issued. A very detailed description how to add device drivers under VxWorks can be found in [16], where this is discussed by way of the example of the serial device driver for the S-DALINAC control system.

F.2 VMEbus drivers under VxWorks for AXPvme

One special feature of the VxWorks kernel is the fact that its I/O system can be extended during run-time. The concept of loadable drivers makes it possible that a device driver

Tab. F.1: The routines which make up the seven driver entry points.

Entry point	Function description
create	A new device is to be created and added to the I/O system.
delete	The device is to be removed from the I/O system.
open	A channel to the device is to be opened. Often the same as the create entry point.
close	A channel to the channel is to be closed. Often the same as the delete entry point.
read	A read access is performed onto the device.
write	A write access is performed onto the device.
ioctl	Special settings for the communication properties are handled by this routine. Applicable only to terminal drivers.

can be added and removed by any application running on the system. The need of kernel re-compilation when adding a new controller is no longer present. This feature proved helpful especially in the development phase since it allowed a rapid prototyping of device drivers.

The unique architecture of the 64 bit wide Alpha CPU and the private bus accesses of the AXPvme SBC must be reflected in the development of VMEbus drivers. The generic VMEbus concept implies that the various VMEbus modules can be seen as an one to one extension to the system controllers memory. The Alpha CPU can access 32 or 64 bit boundaries, so that extra care has to be taken on how to access memory cells with smaller sizes.

The Alpha CPU has no direct access to the VMEbus thus it must map this address space into its internal one over a PCI-bridge interface as shown in Fig. F.1. The PCI-bus is the internal bus of the AXPvme SCB to connect the on board peripherals. The bridge connects the internal CPU-bus (L-bus) with the PCI-bus.

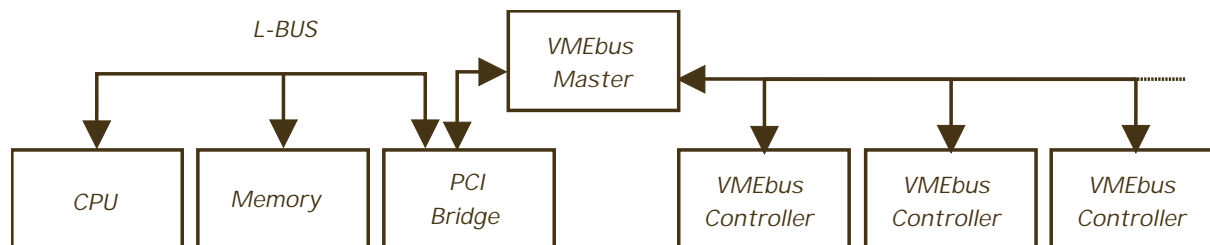


Fig. F.1: Data path from the CPU bus to the VMEbus.

The Board Support Package (BSP) for AXPvme provides a set of functions for a transparent mapping. Device drivers must make use of these routines and cannot assure a

seamless access to the control registers. These specialities are explained in detail in [16].

Usually the appropriate device drivers are bundled with the commercial hardware featuring a ready-to-use controller. Unfortunately there is no manufacturers support for the combination of a SBC equipped with an Alpha CPU and VxWorks. This leads to the situation that all drivers incorporated in the S-DALINAC local control system had to be developed on site.

Appendix G

LCP implementation under VxWorks

G.1 Ethernet hooks

The Linac Control Protocol is one of the key features of the S-DALINAC control system since it connects the remote and local control system. A replacement of this communication protocol would implicate that new soft- and hardware for the remote applications would be mandatory. Based on the fact that it is a private protocol definition not based on the IP standards, the kernel had to be extended with hook functions taking over the handling of the OSI Physical and Data Link Layers [35]. For both communication directions – incoming and outgoing frames – a separate hook function is installed. Those routines bypass the habitual frame processing and the application programmer gets access to any Ethernet frame addressed to the NIC. The NIC, being the link between the kernel software and the hardware of the physical media, by default filters only the manufactures hardware address. Thus the NIC filter specifications were modified. The VxWorks routine `tuFilterSetup` sets the filtering for the attached Ethernet device.

IEEE 802.3 SNAP frame. The data sent as a byte stream over the physical media Ethernet is formatted in so-called frames. The LCP uses the IEEE 802.3 SNAP frame format [25] which is displayed in Fig. G.1. Every frame begins with a preamble which is used for synchronization purposes. It is stripped off by the NIC before the frame is passed to the operating system. Thus for the programmer the frame begins with the six byte destination field (DST) as shown in the figure. The next six byte contain the source address (SRC), which must be unique on the Ethernet LAN. After the length field the next bytes provide a Data Link Service, where Logical Link Control (LLC) information can be included. For that purpose the next 2 bytes form the Service Access Point (SAP). It consists of the 1 byte Destination Service Access Point (DSAP) address, 1 byte Source Service Access Point (SSAP) address and 1-2 byte Control field. The LLC is extended with a 5 byte Sub-Network Access Protocol (SNAP) encapsulation. The first three bytes are reserved for the vendor code (normally the same 3 bytes as the first 3 bytes of the source address) and the next 2 bytes specify the type field. The user data can then contain between 38 and 1492 bytes.

tuFilterSetup. The destination address of any Ethernet frame on the physical media is examined by the NIC according to the current hardware address filter specifications. For

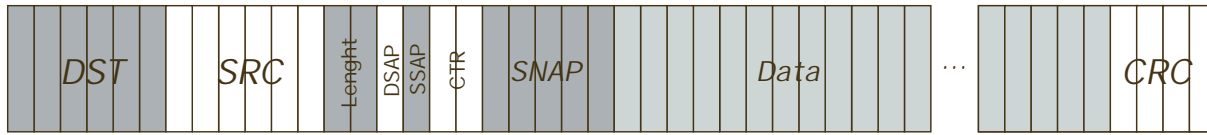


Fig. G.1: Detailed IEEE 802.3 SNAP frame. The preamble bytes are omitted, as they are not accessible. For further explanations see the text.

the LCP implementation the filter must be set up to accept the former RECYC DECnet address and the LCP/ACP multicast address. To still allow IP processing, the filter has to be set up with the NIC manufactures hardware address first. This address can be obtained with the function call `sysEnetAddrGet`. Further parameters to the `tuFilterSetup` function, next to the array holding the MAC addresses, are the unit number of the attached `tu` device (here 0 since the SBC carries only one Ethernet interface) and the type of filtering (here `TU_16_PERFECT`: filters one to 16 addresses without any frame loss).

Input hook. Since the frame is not processed by the operating system, the application programmer must be aware that the incoming hook routine must be as fast as possible, because the NIC's receive buffer is blocked during hook execution. If the buffer is blocked no Ethernet frames addressed to the NIC can be received and may get lost. The input hook routine checks for the source address being a valid and known one and whether the SNAP field belongs to the LCP definition. If this is the case, the frame is copied into the message queue and sent to the dispatch task.

Output hook. This routine is called just before transmission and the whole frame header can be modified. An IEEE 802.3 SNAP header is prefixed and the source address is adjusted to the system's DECnet like MAC address.

Both hook functions have in common that if they have handled the frame, they return `TRUE` to indicate that no further action should be taken. Returning `FALSE` signals to the operating system that normal processing with system routines should be performed. The function prototypes of the hook routines must be compliant to the following syntax:

```
BOOL ethernetHook (struct ifnet *pIf, char *buffer, int length).
```

The function parameters are: `pIf` – a pointer to the interface the packet will be sent on/received from, `buffer` – a pointer to the packet to be transmitted/received, `length` – the length of the packet.

G.2 Protocol extensions

To propagate the new services offered by the local control system, new message types had to be introduced in the LCP. The message types listed in Tab. G.1 are only known to the master IOC application and the *ACP*, the slave IOC and the knob-server do not need to incorporate them since they do not exchange such high level commands.

Tab. G.1: New LCP message types, next to their hexadecimal value and their parameters, a short description of the purpose is provided.

Message type	Value	Parameters	Purpose
MSGT_CYCLE	0x00120000	None	Depending on the source, this message type serves two purposes: When sent from <i>ACP</i> to the IOC it indicates to cycle all magnets that carry the cycle characteristic. When sent from the IOC to <i>ACP</i> , the OPI is informed that a cycling process is over and it can resume execution.
MSGT_CYCLEONE	0x00130000	Controller Name (2 chars), controller number (1 byte), unit number (1 byte), value to set after cycling.	Indicating the command to cycle only one magnet, where the device can be a single one or a grouped one. In the latter case the devices forming the composite are cycled simultaneously.
MSGT_GROUP	0x00140000	Number of devices to group (1 byte), number of slot (1 byte) and then a contiguous concatenation of the combination of device controller name (2 chars), controller number (1 byte) and unit number (1 byte).	The magnets are grouped, with the first specified magnet being the root of the composite device.
MSGT_UNGROUP	0x00150000	Slot number (1 byte)	Demanding the decomposition of a group of devices.
MSGT_CLEARFLG	0x00160000	None	After the restoring of a saved setting, all devices should be considered as cycled.

Appendix H

New ACP commands

This appendix summarizes the new commands in the remote application *ACP* that have been implemented after the functions were provided by the new local control system system. The tables H.1 and H.2 list the purpose and the command syntax of the new *ACP* commands.

Tab. H.1: This table lists the new commands in the *ACP* application and describes their purpose.

Command	Purpose
Group devices	This command sends the group command to the master IOC and removes all group members from the dial-list, except for the root device.
Ungroup devices	This sends the command to the master IOC to decompose the group that is identified by the name of the root device.
Show grouped devices	Displays a list of all groups with their members.
Show touched devices	Displays a list of all quadrupoles and dipoles of which the values have been changed after they have been cycled.
Cycle all	All quadrupoles and dipoles are cycled, whereas the final value for the bending magnets and quadrupoles is zero and for the correction coils is their minimum value. Before the cycling command is sent to the master IOC, the operators are informed to insert the Faraday cup and are asked to approve the command (same with Cycle single device).
Cycle single device	A single device, that can also be a grouped device, is cycled and then set to the final value.

The restore command "61 (load file)" was modified, a previous save setting is now restored in a multistage process. First the operators are asked if they want to cycle all devices before loading the old settings. Then all, except the values for the correction trim coils are restored and after a time delay of 140 s to let the eddy currents dissipate, the trims' values are also restored.

Tab. H.2: This table lists the new commands in the *ACP* application and their parameters.

Command	Command Number	Parameters
Group devices	44	Comma separated list of power supply device names.
Ungroup devices	45	Name of the root device.
Show grouped devices	46	None.
Show touched devices	47	None.
Cycle all	48	None.
Cycle single device	49	Name of device to cycle and final set value.

Glossary

10Base-2	10 Mbit/s Ethernet system based on Manchester signal encoding transmitted over thin coaxial cable. Also called ThinWire or Cheapernet.
ANSI	American National Standard Institute.
AUI	Attachment Unit Interface. The 15-pin signal interface that carries signals between a station and an outboard transceiver.
BNC	Bayonet Navy Connector. A bayonet locking connector used on 10Base-2 thin coaxial cable segments.
Boolean variable	A variable which can have two values either "TRUE" or "FALSE".
Bootp	The initial loader protocol as defined in RFC 951 for loading the system kernel over the network after system start-up.
BSP	Board Support Package. Set of hardware dependent routines.
Client-server model	The model of interaction in a distributed system in which a program at one site sends a request to a program at another site and awaits a response. The requesting program is called the client; the program satisfying the request is called the server.
Common block	Static data structure in RAM, created during compile time. Feature of the operating system RSX.
Compiler	A program that converts another program from its source code into machine code (object code).
CPU	Central Processing Unit. The (usually) single integrated circuit (IC) that does the actual interpreting of program instructions and processing of data in a computer.
Daisy-chain	Bus wiring scheme in which, for example, device A is wired to device B, device B is wired to device C, etc. The last device is normally wired to a resistor or terminator.
Deadlock	A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause.

DECnet	Proprietary Network Protocol developed by DEC.
DELNI	Digital Equipment Local Network Interconnect. A transceiver splitter box that taps into the Ethernet cable. Each DELNI box enables up to eight stand-alone computer workstations to be tied into the Ethernet.
DMA	Direct Memory Access. A technique for transferring data from main memory to a device without passing it through the CPU.
EPICS	Experimental Physics and Industrial Control System. A set of toolboxes used to build up a control system.
EPROM	Erasable Programmable Read Only Memory. A reusable memory chip that can be programmed electrically and erased by exposure to ultraviolet light. (Also called Electrically Programmable ROM).
Ethernet	Networking system invented 1973 by B. Metcalf, Xerox Parc, and developed by a consortium of Digital, Intel and Xerox.
GDB	GNU debugger.
Global section	(RSX,VMS) A section in the main memory which can be accessed by all processes in the system.
GNU	"GNU's Not Unix". The GNU project develops a complete Unix-like operating system which is free software.
GUI	Graphical User Interface.
Hook	Runtime kernel extension.
IEEE	Institute of Electrical and Electronics Engineers.
IC	Integrated Circuit.
IOC	Input/Output controller. The combination of an industrial bus system with a Single Board Computer.
IP	Internet Protocol. Networking protocol for wide area networks. Today used in LAN and WAN.
ISO	International Standard Organization responsible for setting open, vendor-neutral standards and specifications for items of technical importance.
Kernel	The central module of an operating system, that loads first and remains permanently in main memory. It is responsible for memory, process, and disk management.
LAN	Local area network. A communications network linking a number of computers in a relatively small geographical (i.e., "local") region.

MAC address	Media Access Control address. Layer 2 protocol address by which a NIC is uniquely identified in a network. This address is worldwide registered by the IEEE-SA.
Message queue	A mechanism for synchronous or asynchronous interprocess communication.
Multi-tasking	The ability to execute more than one task at the same time. Only one CPU is involved, but it switches from one task to another so quickly that it gives the appearance of executing all of the tasks at the same time.
NIC	Network Interface Card. A set of electronics that provides a connection between a computer and a LAN.
OpenVMS	Proprietary operation system developed by Digital Equipment, now Compaq.
Operation system	Software that controls the execution of computer programs; may provide a range of services as well (data management, debugging, scheduling, etc.).
OPI	Operator Interface. A remote application providing the man-machine interface to the local control system.
OSI	Open Systems Interconnect. Reference model developed by ISO to provide a common organizational scheme for network standardization efforts.
PAL	Programmable Array Logic, IC programmed with custom logic equations.
PCI	Peripheral Component Interconnect. A standard for interconnecting peripherals to a personal computer designed by Intel.
POSIX	Definition for a Portable Operating System, and the "IX" to have it sound more like UNIX. Standard 1003.1 and 1003.4 are real-time extensions.
Q-bus	Proprietary system bus for connecting interface cards, CPU and memory developed by Digital Equipment.
Real-time	Pertaining to the processing of data by a computer in connection with another process outside the computer according to time requirements imposed by the outside process.
RFC	Short for Request for Comments, a series of notes about the Internet, started in 1969. An RFC can be submitted by anyone and eventually evolves into an Internet standard.

RPC	Remote Procedure Call. Protocol definition to execute a command on another host.
rsh	Remote Shell. Unix-command for executing a shell on a remote computer.
RSX	Proprietary operation system developed by Digital Equipment, now owned by Compaq.
SAP	Service Access Point. Protocol fields in an Ethernet frame as defined in IEEE 802.2 which define the high-level protocol that the data in the frame is intended for (Destination SAP, DSAP) or coming from (Source SAP, SSAP)
SBC	Single Board Computer. One printed circuit contains all electronics required for a fully operational computer.
Scheduler	Kernel program that distributes the resource CPU among the processes in the system.
Semaphore	A Boolean variable that is only accessed by two indivisible operations, "Give" and "Take".
Shell	Primary interface between a user and the operating system.
SNAP	Sub-Network Access Protocol. Protocol fields in an Ethernet frame as defined in IEEE 802.2 for carrying older protocol type identifier.
Starvation	A process enters starvation if it is permanently blocked from accessing a resource by other processes
Task	A task represents the execution of a sequential program or a sequential component of a concurrent program. Each task deals with a sequential thread of execution, there is no concurrency within a task.
TCP	Transmission Control Protocol. The most common Transport Layer protocol used on Ethernet and the Internet. TCP is built on top of Internet Protocol (IP); it adds reliable connection-oriented communication.
tftp	Trivial File Transfer Protocol. Protocol defined in RFC 1350 to download an image from a server.
Thread	Single, sequential flow of control within a program with a single point of execution.
Transceiver	Combination word of transmitter and receiver. A set of electronics that sends and receives signals on an Ethernet media system.

Trackball	A pointing device consisting of a ball housed in a socket – like an upside-down mouse.
UDP	User Datagram Protocol. UDP is build on top of Internet Protocol (IP); it adds a connectionless communication providing simple but unreliable datagram services.
UNIX	Operating system developed by AT&T. Many variants exist (TRU64 from Compaq, IRIX from SGI, Solaris from SUN etc.)
VMEbus	(VersaModule Eurocard bus) A 32-bit bus developed by Motorola, Signetics, Mostek and Thompson CSF. It is widely used in industrial, commercial and military applications with over 300 manufacturers of VMEbus products worldwide. VME64 is an expanded version that provides 64-bit data transfer and addressing.
VxWorks	A real-time multi-tasking operating system from Wind River Systems.
WAN	Wide Area Network. A network which extends over large distances and is normally supported by common carrier transmission services.

Bibliography

- [1] F. Gudden, G. Fricke, H.-G. Clerc und P. Brix, Z. Phys. **181** (1964) 453.
- [2] A. Richter, Proc. 5th Europ. Part. Acc. Conf., Barcelona, 1996, ed. by S. Myers et al., (IOP Publishing, Bistol, Philadelphia, 1996), 110.
- [3] V. Huck, Dissertation, Technische Hochschule Darmstadt, 1992, D17.
- [4] J. Horn, Dissertation, Technische Hochschule Darmstadt, 1996, D17.
- [5] D. Flasche, Dissertation, Technische Hochschule Darmstadt, 1989, D17.
- [6] L. Dalesio, J. Hill, M. Kraimer, D. Murray, S. Hunt, M. Claussen, C. Watson, J. Dalesio, *The Experimental Physics and Industrial Control System Architecture*, (ICALEPCS, Berlin, Germany, 1993).
- [7] J. Mühl, Diplomarbeit, Institut für Kernphysik, Technische Universität Darmstadt, 2000, unpublished.
- [8] V. Huck, Diplomarbeit, Institut für Kernphysik, Technische Hochschule Darmstadt, 1987, unpublished.
- [9] H. Weise, Dissertation, Technische Hochschule Darmstadt, 1993, D17.
- [10] J. Pinkow, Diplomarbeit, Institut für Kernphysik, Technische Hochschule Darmstadt, 1990, unpublished.
- [11] M. Hertling, Diplomarbeit, Institut für Kernphysik, Technische Universität Darmstadt, in preparation.
- [12] H. Schmidt, Diplomarbeit, Institut für Kernphysik, Technische Hochschule Darmstadt, 1984, unpublished.
- [13] W. Löw, Dissertation, Technische Hochschule Darmstadt, 1983, D17.
- [14] K. Alrutz-Ziemssen, Dissertation, Technische Hochschule Darmstadt, 1990, D17.
- [15] H.-D. Gräf, H. Miska, E. Spamer, O. Titze and T. Walcher, Nucl. Istr. and Meth. **153** (1978), 9.
T. Walcher, R. Frey, H.-D. Gräf, E. Spamer and H. Theissen, Nucl. Istr. and Meth. **153** (1978), 17.

- D. Schüll, J. Foh, H.-D. Gräf, H. Miska, R. Schneider, E. Spamer, H. Theissen, O. Titze and T. Walcher, Nucl. Instr. and Meth. **153** (1978), 29.
 J. Foh, R. Frey, R. Schneider, D. Schüll, A. Schwierczinski, H. Theissen and O. Titze, Nucl. Instr. and Meth. **153** (1978), 43.
- [16] M. Platz, Diplomarbeit, Institut für Kernphysik, Technische Universität Darmstadt, 1999, unpublished.
 - [17] S. Watzlawik, Dissertation, Technische Universität Darmstadt, in preparation.
 - [18] R. Eichhorn, Dissertation, Technische Universität Darmstadt, 1999, D17.
 - [19] H. Gomaa, *Software design methods for concurrent and real-time systems*, (SEI Series in Software Engineering, Addison-Wesley, Reading/MA., 1996).
 - [20] A. Tanenbaum, *Modern Operating Systems*, (Prentice-Hall Inc., London, 1992).
 - [21] *Information Technology-Standardized Application Environment Profile – POSIX Realtime Application Support (AEP)*, ISO/IEC ISP 15287-2; IEEE Std **1003.13** 3-124, 2000.
 - [22] VxWorks, Wind River Systems Inc., <http://www.wrs.com>.
 - [23] *IEEE standard for a versatile backplane bus: VMEbus*, ANSI/IEEE Std **1014**, 1987.
 - [24] W. Petterson, *The VMEbus Handbook*, VITA publication, 1999.
 - [25] *Information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements. Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access method and physical layer specifications LAN MAN Standards*, IEEE Std **802.3**, 1998.
 - [26] *jlab controls*, <http://www.jlab.org/accel/controls/controls.html>.
 - [27] *DOOCS, The Distributed Object Oriented Control System*, <http://tesla.desy.de/doocs/doocs.html>.
 - [28] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns, Elements of Reusable Object-Oriented Software*, (Addison-Wesley Professional Computing Series, Reading/MA., 1995).
 - [29] D. Parnas, *On the Criteria on Decomposing a System into Modules*, (Communications ACM, 1972).
 - [30] B. Stroustrup, *C++ The Programming Language*, (Addison-Wesley, Reading/MA., 1997).

- [31] J. Gosling, H. McGilton, *The Java Language Environment – A white paper*, (Sun Microsystems Inc., Mountain View/CA., 1995).
- [32] A. Stiller, private communication, 1996.
- [33] B. Oestereich, *Objektorientierte Softwareentwicklung*, (R. Oldenbourg, München, 1998).
- [34] H. Wiedemann, *Particle Accelerator Physics*, (Springer, Berlin, 1993).
- [35] J.D. Day, H. Zimmermann, *The OSI Reference Model*, (Proc. of the IEEE, **71**, 1983), 1334-1340.
- [36] U. Laier, Diplomarbeit, Institut für Kernphysik, Technische Universität Darmstadt, 1998, unpublished.
- [37] M. Platz, Dissertation, Technische Universität Darmstadt, in preparation.
- [38] C. Lüttge, Dissertation, Technische Hochschule Darmstadt, 1994, D17.
- [39] K. Etschberger, *CAN Controller-Area-Network*, (Hanser, München, 2000).
- [40] Th. Winkler, Diplomarbeit, Institut für Kernphysik, Technische Hochschule Darmstadt, 1993, unpublished.
- [41] *S-DALINAC Handbook*, <http://linnt2.ikp.physik.tu-darmstadt.de/S-DALINAC>.
- [42] F. Hofmann and Y. Kalmykov, private communication, 2000.
- [43] T. Wesp, Dissertation, Technische Universität Darmstadt, 1998, D17.
- [44] *AXPvme Single-Board Computer Installation Guide*, (Digital Equipment Corporation, Manyard/MA., 1993).
- [45] E. Dijkstra, Co-operating Sequential Processes, in F. Genuys (ed), *Programming Languages*, (Academic Press, New York, 1968) 43-112.
- [46] G. Pajari, *Writing UNIX Device Drivers*, (Addison-Wesley, Reading, 1992).

Danksagung

An dieser Stelle möchte ich all denen meinen Dank aussprechen, die zum Gelingen dieser Arbeit beigetragen und mich in den Jahren meiner Diplom- und Doktorarbeit begleitet haben.

An erster Stelle möchte ich Herrn Professor Dr. Dr. h.c. mult. A. Richter für die Stellung des Themas und sein stetiges Interesse am Fortgang dieser Arbeit danken. Das Vertrauen, das er in mich gesetzt hat, mir eine Arbeit im Grenzbereich zwischen Physik und Informatik anzuvertrauen, hat mich immer wieder angespornt. Ganz besonders danken möchte ich ihm für die Ermöglichung der Teilnahme an der *Cern School of Computing (1997)* und den zahlreichen Konferenzen und Tagungen.

Herrn Professor Dr.-Ing. T. Weiland danke ich für die Bereitschaft, das Korreferat zu übernehmen.

Für die besonders freundschaftliche Betreuung in all den Jahren bin ich Herrn Dr. O. Titze zu Dank verpflichtet. Sein unermüdlicher Einsatz und die stete Diskussionsbereitschaft waren immer ein wichtiger Rückhalt. Die Vermittlung von vielfältigen Kontakten hat viel zum Gelingen dieser Arbeit beigetragen.

Ohne die detaillierte und umfangreiche Erläuterung des Betriebs des S-DALINAC wäre die erfolgreiche Umsetzung der neuen Anforderungen in das Kontrollsystem nicht möglich gewesen. Hier sei der ganzen Beschleunigergruppe, insbesondere Herrn Dr. H.-D. Gräf, mein herzlicher Dank ausgesprochen.

Die Diplomarbeiten von Frau Dipl.-Phys. J. Mühl und den Herren Dipl.-Phys. M. Platz und cand.-phys. M. Hertling sind ein wertvoller Beitrag zu dieser Arbeit.

Dank des überdurchschnittlichen Einsatzes der Herren Gheybi und Dipl.-Ing. U. Bonnes wurden Probleme mit der Hardware und Hardwareentwicklung immer schnellstens gelöst.

Mein Dank gilt auch allen bisher nicht namentlich genannten Mitgliedern der Arbeitsgruppe für die kollegiale Zusammenarbeit und das außerordentlich gute Arbeitsklima.

Meinen Eltern danke ich für das immerwährende Vertrauen und die liebevolle Unterstützung. Ein besonderer Dank sei auch meinem Lebensgefährten Mathias ausgesprochen, der mir, besonders in der Endphase dieser Arbeit tatkräftig zur Seite stand.

Diese Arbeit wurde unterstützt durch Mittel des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 06 DA 665 I und der Deutschen Forschungsgemeinschaft unter den Förderkennzeichen Ri 242/12-1 und Ri 242/12-2.

Lebenslauf

Simone Georgia Richter

13. Dezember 1969	Geboren in Berlin
1975 – 1976	Besuch der Volksschule in Nürnberg-Katzwang
1976 – 1979	Besuch der Verbands-Volksschule Fischbach
1979 – 1988	Besuch des Göttenbach-Gymnasiums
Juni 1988	Abitur
1988 – 1996	Studium der Physik an der Technischen Hochschule Darmstadt mit Abschluß Diplom
Januar 1996	Abschluß des Studiums mit Diplom
Januar – Juni 1996	Wissenschaftliche Hilfskraft am Institut für Kernphysik der Technischen Universität Darmstadt
seit Juli 1996	Wissenschaftliche Mitarbeiterin am Institut für Kernphysik der Technischen Universität Darmstadt

Eidesstattliche Erklärung:

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertation selbständig verfaßt, keine anderen als die angegebenen Hilfsmittel verwendet und bisher noch keinen Promotionsversuch unternommen habe.

Darmstadt, im Dezember 2000

